The Dissertation Committee for Jeremy Thomas Murphy Certifies that this is the approved version of the following dissertation:

Patent-Based Analogy Search Tool for Innovative Concept Generation

Committee:

Kristin Wood, Supervisor

Joseph Beaman

Matthew Campbell

Richard Crawford

Dan Jensen

Patent-Based Analogy Search Tool for Innovative Concept Generation

by

Jeremy Thomas Murphy, B.S.M.E; M.S.E

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2011

Dedication

To my loving wife Jamie, whose patience and understanding knows no bounds.

Patent-Based Analogy Search Tool for Innovative Concept Generation

Publication No._____

Jeremy Thomas Murphy, Ph.D.

The University of Texas at Austin, 2011

Supervisor: Kristin Wood

Design-by-Analogy is a powerful tool to augment the traditional methods of concept generation and offers avenues to develop innovative and novel design solutions. Few tools exist to assist designers in systematically seeking and identifying analogies from within design repositories such as the United States Patent and Trademark Office patent database. A new tool for extracting functional analogies from patents has been developed to perform this task utilizing a Vector Space Model algorithm to quantitatively evaluate the functional similarity between design problems and patent descriptions of products.

Initially, a Boolean Search approach was evaluated and several limitations were identified such as a lack of quantitative metrics for determining search result relevancy ranking as well as inadequate query mapping methods. Next, a Vector Space Model search tool was developed which includes extensive expansion of the Functional Basis using human-based term classification and automated document indexing techniques. The

resulting functional patent controlled vocabulary consists of approximately 2,100 unique functions extracted from 65,000 randomly selected patents. The patent search database was generated by indexing 275,000 patents selected from the over 4 million patents available in digital form.

A graphical user interface was developed to facilitate query vector generation, and the accompanying search result viewing interface provides data clustering and relevancy ranking. Two case studies are conducted to evaluate the efficacy of the search engine. The first case study successfully replicated the functional similarity results of a classic Design-by-Analogy problem of the guitar pickup winder. The second case study is an original design problem consisting of an automated window washer, and the results illustrate the range of analogically distant solutions that can be extracted ranging from very near-field, literal solutions to the far-field cross domain solutions.

Finally, the search tool's efficacy with regard to increasing quantity and novelty of ideas produced during Concept Generation is experimentally evaluated. The two factors evaluated are first whether analogies improved performance and second how the functionality level of the analogy impacted performance. The experimental results showed an increase in novelty for high functionality analogies compared with the control and other experimental groups. No statistically significant difference was found with regard to quantity of ideas generated.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

INTRODUCTION TO CONCEPTUAL DESIGN

The product design process consists of three distinct phases: the problem definition phase, the conceptual design phase and the embodiment design phase (Ullman, 2003, Otto and Wood, 2001). The definition phase identifies an opportunity through market and customer needs analysis which are then mapped to a set of engineering specifications. During the conceptual design phase, concepts are generated to meet the needs of the design problem identified in the previous phase. The concepts generated are evaluated using a variety of concept selection tools, and the "best" or preferred design is selected for implementation. The embodiment design phase consists of tasks that iteratively define the details of the final embodiment of the design such as geometry, material properties, architecture and performance. The product design process is iterative and previous phases are repeated in order to deliver a product, or portfolio of products, that meets or exceeds the customers' needs and expectations.

The conceptual design phase and specifically the concept generation process is a critically important phase of the product development process. The concepts generated ultimately establish the architecture and performance of the final design (Pahl and Beitz, 1996). This phase also establishes much of the project cost (Römer, Weißhahn and Hacker, 2001). At the beginning of concept generation, the solution design space is entirely open, and the greatest opportunities for innovation are available. Several creative techniques are employed to generate design concepts such as brainstorming, brainsketching, and the CSketch/6-3-5 method (Osborn, 1957, Vangundy, 1988, Otto and Wood, 2001, Markman and Wood, 2009). Complementary to these methods, the use of functional analogies has been promoted as an important technique for synthesizing

innovative and novel solutions, and research has empirically verified this effect (Cagan, et al., 2011, Chan, et al., 2011, Linsey, et al., 2006). Appropriate representations of the design problem must first be developed to enable the effective application of these concept generation techniques. The following section describes functional representations of design problems in detail.

Design Problem Representation and Functional Modeling

Designers use various external design representations for a wide range of functions and it has been shown that specific representations are better suited for specific tasks e.g. sketching for solution development, complex models for verifying requirements, etc (Römer, et al., 2001). For concept generation, design problems should be represented by a set of solution-neutral functions to minimize design fixation and enable the greatest number of concepts to be considered (Pahl and Beitz, 1996, Chakrabarti and Bligh, 2001, Otto and Wood, 2001). The process of developing a functional representation of a concept is called functional modeling and begins with an abstracted black box formulation of the overall product function. The black box model is then decomposed it into sub-functions connected by their associated input flows.

Decomposing the abstracted functional representation into aggregated function chains results in the creation of a repeatable function structure representing the internal functionality of a concept (Kurfman et al, 2001, 2003). In addition to the development of a decomposition methodology, extensive efforts have produced a standard language for representing the function and flows associated with each sub-function called the Functional Basis (Stone and Wood 2000a, Hirtz et al, 2002) as shown in Table 1.

Table 1: Reconciled Functional Basis functions

Class (Primary)	Secondary	Tertiary	Correspondents
Branch	Separate		isolate, sever, disjoin
		Divide	detach, isolate, release, sort, split, disconnect, subtract
		Extract	refine, filter, purify, percolate, strain, clear
		Remove	cut, drill, lathe, polish, sand
	Distribute		diffuse, dispel, disperse, dissipate, diverge, scatter
Channel	Import		Form entrance, allow, input, capture
	Export		dispose, eject, emit, empty, remove, destroy, eliminate
	Transfer		carry, deliver
		Transport	advance, lift, move
		Transmit	conduct, convey
	Guide		direct, shift, steer, straighten, switch
		Translate	move, relocate
		Rotate	spin, turn
		Allow DOF	constrain, unfasten, unlock
Connect	Couple		associate, connect
		Join	assemble, fasten
		Link	Attach
	Mix		add, blend, coalesce, combine, pack
Control Magnitude	Actuate		enable, initiate, start, turn-on
	Regulate		Control, equalize, limit, maintain
		Increase	allow, open
		Decrease	close, delay, interrupt
	Change		adjust, modulate, clear, demodulate, invert, normalize, rectify, reset, scale, vary, modify
		Increment	amplify, enhance, magnify, multiply
		Decrement	attenuate, dampen, reduce
		Shape	compact, compress, crush, pierce, deform, form
		Condition	prepare, adapt, treat
	Stop		end, halt, pause, interrupt, restrain
		Prevent	disable, turn-off
		Inhibit	shield, insulate, protect, resist
Convert	Convert		Condense, create, decode, differentiate, digitize, encode, evaporate, generate, integrate, liquefy, process, solidify, transform
Provision	Store		accumulate
		Contain	capture, enclose
		Collect	absorb, consume, fill, reserve
	Supply		provide, replenish, retrieve
Signal	Sense		feel, determine
		Detect	discern, perceive, recognize
		Measure	identify, locate
	Indicate		announce, show, denote, record, register
		Track	mark, time
		Display	emit, expose, select
	Process		compare, calculate, check
Support	Stabilize		steady
	Secure		constrain, hold, place, fix
	Position		align, locate, orient

The Functional Basis consists of a set of function and flow words used as a verb-object couple to describe the action imparted on the input flows of each sub-function and is sufficiently abstracted to cover the entire function and flow space. This basis forms a standard taxonomy for describing a design concept and enables concepts or products to be directly compared. Figures 1and 2 illustrate the functional decomposition and modeling process for a portable jigsaw product. The black box representation in Figure 1 defines the system level functionality as well as the input and output flows to the system.

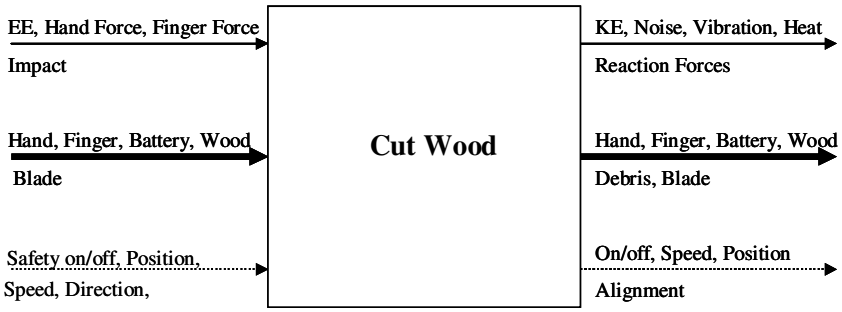


Figure 1: Functional model at system level for a Black and Decker Firestorm jigsaw.

In Figure 2, the system is further decomposed to an atomic functional level. The atomic functional level is defined as the level where further decomposition is not possible or adds no information to the model. For each input flow, the designer must visualize the operations and transformations which take place internally to the system through to the outputs. These operations are expressed in the verb-object form and arranged in function chains representing a logical causal relationship from input to output.

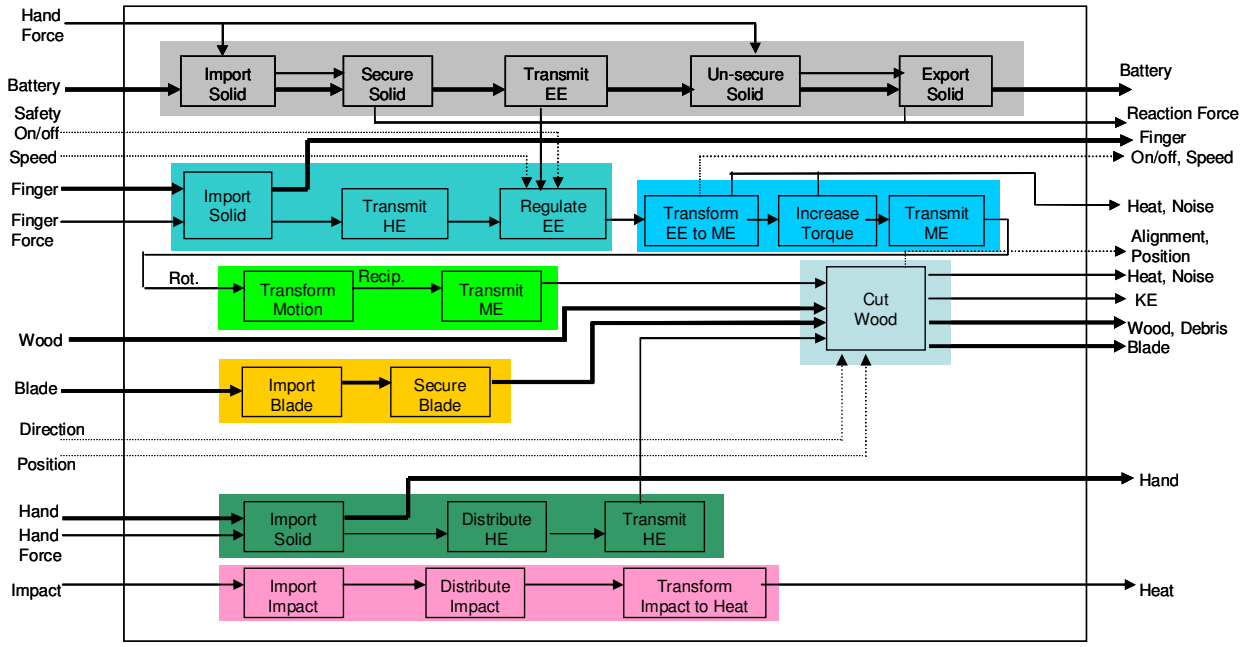


Figure 2: Functional model at the atomic functional level for a Black and Decker Firestorm jigsaw.

Additional methods can be applied to the functional model to identify modules and interface boundaries (Stone and Wood, 2000b). This information can be used to simplify a complex functional model as well as discover opportunities to improve manufacturability, maintainability, and reliability early in the design process through function sharing and proper interface design. The modules contained in the jigsaw product are signified by the function chains contained in each colored box in Figure 2.

A standardized functional model also facilitates archiving and retrieval of design knowledge. To that end, several systems have been developed to store the design knowledge contained in the functional models for design reuse (Bohm and Stone, 2004a, 2004b, 2005, Szykman et all, 1998, 2000). In addition, computational tools have been developed to exploit the knowledge contained in the design repositories for the purpose

of concept variant generation (Bryant et al, 2005a, 2005b; Terpenny and Mathew, 2004, Potter et al, 2003).

One limitation of the functional modeling procedure is that selection of flow variables necessitates that process choices must be made early in the conceptual design process. In the jigsaw example, a process choice is made to utilize a battery as the power source. This is an obvious choice for a portable device, but alternative power sources such as fuel cells, solar cells, or pneumatically powered devices are expressly excluded from consideration. The single, domain-dependent model can lead to missed opportunities for novelty and innovation. The benefits of a domain-independent, function-only representation of the design problem are discussed in the context of functional analogies in the following sections.

INTRODUCTION TO ANALOGY

Understanding the cognitive process involved with forming analogies is fundamental to the development of any tool or methodology that seeks to improve the Conceptual Design process. Analogy can be viewed as a mapping of knowledge from one situation (source) to another (target) enabled by a supporting system of relations or representations between situations (Chiu, 2003; Gentner, 1983; Falkenhainer et al, 1989). Two levels of relations exist between potential sources and targets. The first is superficial similarity which refers to the resemblance between source and target. In the design space, superficially similar target would resemble the source architecturally and/or geometrically and operate within the same physical domain. This superficial level of analogy is also referred to as near-field analogy (Cagan, et al., 2011, Schunn, et al., 2011, Chan, et al., 2011). The second level is structural similarity which refers to the underlying resemblance of relations between the objects of the source and the objects comprising the

target (Blanchette and Dunbar, 2000). Forbus, *et al.* (1994) states that structural similarity exists if the relations holding between the objects in the source are similar to the relations between the objects in the target, independently of the similarity between the objects themselves. In the language of functional modeling, the objects are the flows and the relations between the objects are the functions, and by inference, in the design space, structural similarity is equivalent to functional similarity. Targets with structural similarity with little or no superficial similarity are referred to as far-field analogies (Cagan, et al., 2011, Schunn, et al., 2011). Several authors have stated structural (functional) similarity is a critical feature of analogy (Falkenhainer et al, 1989, Gentner, D., 1983, Gentner, D. and Markman, 1997).

In Figure 3, the two levels of analogy are illustrated using examples of devices designed to *transport humans* off-road, where the system level functions and flows are *transport* and *human,* respectively. The qualifier *off-road* also implies a set of secondary input flows such as vibration, uneven terrain, variable surface materials, etc.

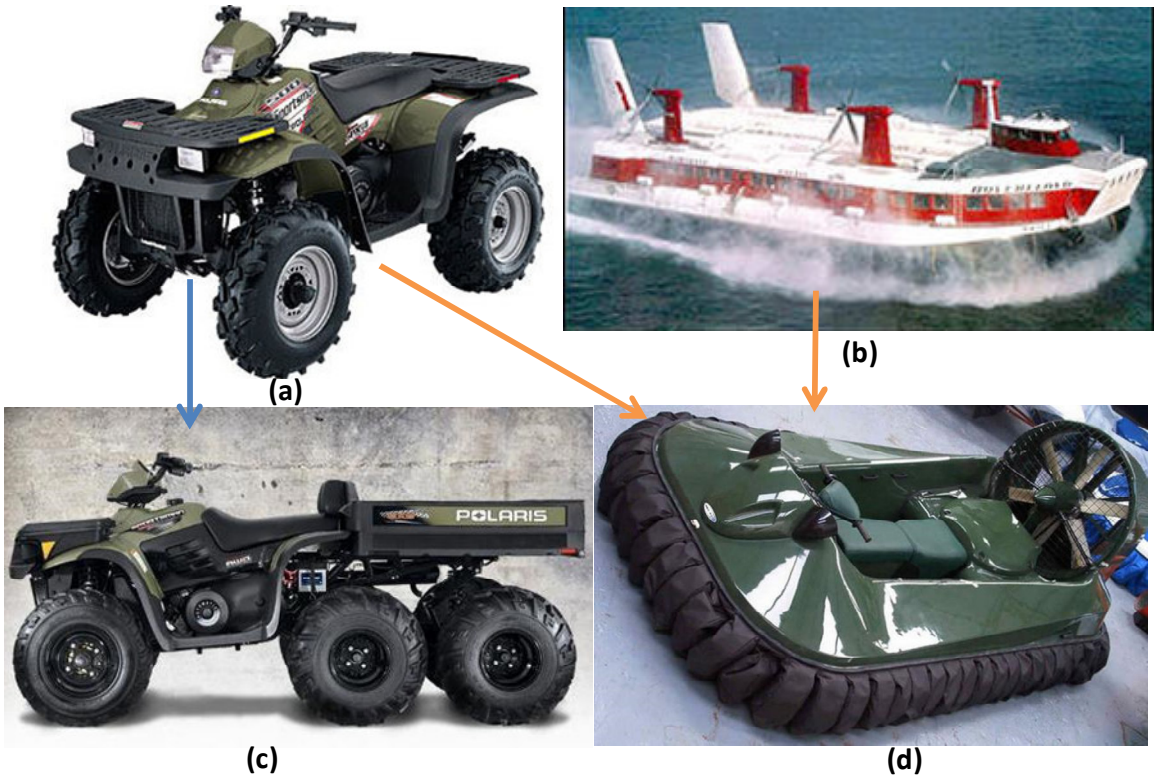


Figure 3: Analogous solutions for *Transport Human* off-road (a) Benchmark solution: four-wheeled ATV (Polaris Sportsman 500, 2003), (b) *Transport* function analogy: English Channel hovercraft (SRN4 Hovercraft, 2000), (c) Superficial target, near-field analogy: six-wheeled ATV (Polaris Sportsman 6x6, 2007), (d) Structural target, far-field analogy: hovercraft (Personal Hovercraft, 2010).

The benchmark solution to this design problem is the four-wheeled all-terrain vehicle (ATV) depicted in Figure 3(a). Using this as the source solution, the superficially similar design concept of the six-wheeled ATV (Figure 3(c)) can readily be derived. The operating principles and domain are identical as are the basic components. The concept would be considered a parametric redesign where the parameter is number of wheels most likely being driven by a load capacity requirement. The structurally similar target concept of the hovercraft (Figure 3(b)) performs the same *transport human* in a different solution domain which is cross the English Channel. The operating principles of

transportation are fundamentally different between the two concepts (wheels versus air cushion) although some superficial similarity does exist in the human/machine interface architecture. Mapping both the benchmark solution and the functional analogy solution to the *Transport Human* off-road domain results in a hybrid solution such as the personal off-road hovercraft shown in Figure 3(d).

This process of analogical comparison fosters new inferences and promotes construing problems in new insightful ways. The potential for creative problem solving is most noticeable when the situation domains are very different (Gentner and Markman, 1997, Cagan et al., 2011). An examination of the spontaneous use of analogy with engineers found that experts use more analogies than novices. Experts also tend to use more generalized analogies as opposed to the more case-based reasoning used by novices (Bell et al, 2004). This difference can be explained because novices have more difficulty retrieving relevant information when needed and have more difficulty mapping concepts from different domains due to a lack of experience (Kolodner, 1997). A structured Design-by-Analogy methodology would be useful for minimizing the effects of the experiential gap between novice and expert. The cognitive analogical process is based on the representation and processing of information, and therefore can be implemented systematically given an appropriate representation of the information and information processing tools (Kryssanov et al, 2001; Goldschmidt and Weil, 1998).

DESIGN-BY-ANALOGY METHODS AND TOOLS

As discussed previously, concept generation is critical to the development of a successful design process. Only relying on a design team's own experiences during concept generation can result in the exclusion of a vast array of feasible concepts especially concepts that lie outside the stated problem domain. Few formal methods exist

to assist designers during the conceptual design phase, but one method which has great potential to produce innovative designs is Design-by-Analogy. Additionally, previous research has shown usage of analogy can mitigate the effects of design fixation (Linsey, et al., 2010).

A robust Design-by-Analogy methodology enables designers to identify non-obvious analogous solutions even in cases where the mapping between concepts is tenuous and/or the concepts occupy different domains. Analogous concepts can be identified by creating abstracted functional models of concepts and comparing the similarities between their functionality. In Figure 4, an innovative design for an electric guitar pickup winder was created using the non-obvious analogy of a vegetable peeler. The abstracted functional model for the pickup winder identified the similarity to the vegetable peeler as well as more obvious analogies to a fishing reel and sewing machine bobbin winder (McAdams and Wood, 2002). Appropriate functional representation of design concepts is as critical to the successful implementation of Design-by-Analogy as is developing a systematic approach to search for and evaluate the utility of functionally similar concepts.

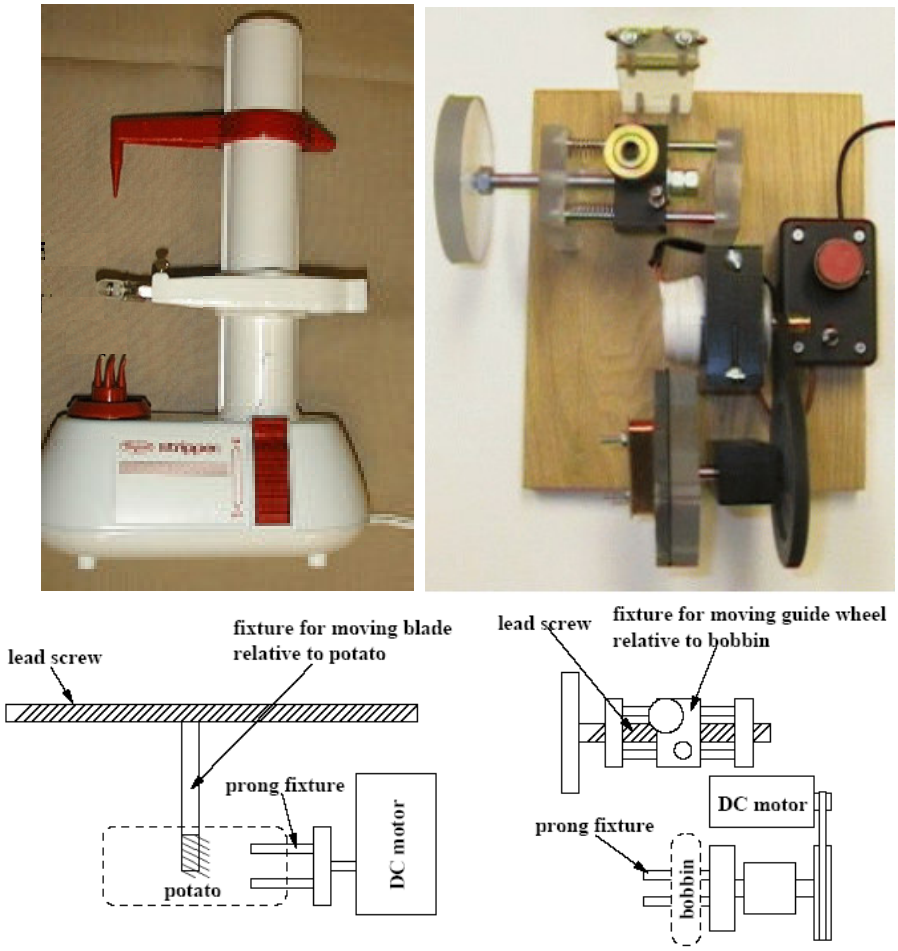


Figure 4: Non-obvious design analogy between an electric vegetable peeler and a guitar pickup winder as a result of functional model comparison. (McAdams and Wood, 2002).

Another structured approach which utilizes biomimetic principles for generating concepts and provides a systematic process for identifying analogous biological phenomena (Hacco & Shu, 2002). Using a semantic representation of the functional requirements problem, keywords are derived that relate the function to biological processes. A search is then preformed using a biology textbook as the reference database.

Several tools have been developed to perform data retrieval and information interpretation using semantic representation (Velardi et al, 1991). ConceptNet is a tool

that implements a semantic representation algorithm to derive contextual meaning and analogies from common natural language phrases (Liu and Sign, 2004a, 2004b). The Semantic Knowledge Representation (SKR) project utilizes semantic representations to enhance information retrieval in large medical databases (Rindflesch and Aronson, 2002; Rindflesch, 1996). The SKR project utilizes a metathesaurus of medical terms to generate a large number of related search terms based on a simple semantic phrase to ensure a greater portion of the database information is queried (Aronson et al, 1994). A similar controlled vocabulary of engineering terms would be useful for generating functional descriptions for analogy searches.

Domain-Independent Functional Semantic Representation with Verbs

Linguistics research states verbs are inherently relational by nature and impose fewer psychological constraints compared to nouns (Gentner, 1981). Verbs represent relational concepts whereas nouns are object-reference concepts. In the following section, an alternative functional representation for design problems is proposed to leverage the cognitive flexibility of the verb.

As discussed previously, functional modeling using the Function-Flow Basis is a useful tool for representing design intent, but the specification of the flow inputs has the consequence of defining the solution domain as a function of the process choices. A truly solution-independent representation should not fix the design space within a particular domain. This can be accomplished by removing the flow objects from the verb-object functional model. The resulting abstract verbal representation is entirely conceptual, relational and solution independent. For example, the functional semantic representation of the jigsaw in Figure 2 can be expressed as:

Import : Secure : Release : Transmit : Regulate : Transform : Distribute

The representation is greatly simplified as common functions acting on different flows collapse into a single verb. It is acknowledged that the functional semantic representation expressed in abstracted basis functions lacks granularity necessary to be useful for concept generation (Chiu, and Shu, 2007). The representation scheme will use lower level functional (tertiary and correspondents) to specify the design problem. The jigsaw translated into one possible combination of the correspondent functions from Table 1 would be represented as:

Capture : Fix : Release : Convey : Control : Transform : Disperse

In the research that follows, an expanded controlled vocabulary of lower level functions is derived, building on the Functional Basis, that spans the entire design space of the design repository.

PATENT DATABASE AS A DESIGN REPOSITORY

Beginning in 1976, the United State Patent and Trademark Office (USPTO) has electronically archived full text versions of all granted patents (www.uspto.gov/patents). As a result, the USPTO patent database now contains over 4 million patents containing a vast amount of embedded design information. In addition to the sheer volume of information contained in the patent database, two fundamental properties of patents, utility patents in particular, make them ideal sources to obtain analogies and concepts that lead to innovative solutions (USPTO, 2010). In order to be patentable, the device or process must be both *useful* and *novel*, where *useful* is defined as being functional and operable. *Novel* is defined as being non-obvious and having not previously existed in the public domain (USPTO, 2010). Another useful feature of the patent database for design information retrieval is the Patent Classification structure. Approximately 450 well-defined primary classification categories have been established to organize and group

patents according to the field of invention. An example of a patent class and definition would be Class 84: *Music* which is defined as follows:

> This class includes the instruments used in producing music and includes (1) electrical music instruments, (2) automatic instruments, and (3) those hand played. The automatic instruments and the hand played instruments have a parallel classification so far as seems practical, and in both the patents are divided in the usual way into four groups, stringed, wind, rigid vibrators, and membranes. Then follow details or features common to groups (1), (2), and (3). This class also includes some accessory devices generally recognized as belonging to the art or industry. (http://www.uspto.gov/web/patents/classification/uspc084/defs084.htm)

The classification system is a powerful element that benefits information retrieval by enabling data clustering for more efficient search result presentation and organization (Kang, et al., 2007). Patents are structurally well formed with distinct partitions, and the sections which contain the embedded design information are the abstract, claims and description. The regular structure of the documents will enable relatively simple implementation of natural language processing techniques to extract functional information. A review of patent search and information extraction literature exposed a dearth of literature on function extraction and concept generation from patents in general. The majority of literature is related to the topics of patent invalidity searches and patent informatics (Trippe, 2003, Tseng, et al., 2007), but the same information extraction principles will be applied for deriving the patent functionality. The detailed procedure is discussed in the following chapters.

HYPOTHESIS AND OBJECTIVES

Objectives

The primary objective of this research is to develop appropriate tools to enable web-based search for design analogies. With these tools, the designer will be able to

methodically search the vast amount of design information available online in patent archives. The resulting analogous concepts will be used to complement and infuse the concept generation process by introduction of non-obvious analogies resulting in innovative conceptual designs. The current work utilizes previous work encompassing functional modeling and representation of design concepts, online information retrieval from text-based databases, and concept similarity metrics to develop a systematic method for extracting near and far-field analogies based on functional similarity. Experimental verification will be conducted to verify the efficacy of the analogy search tool.

Hypothesis

Based on the prior literature and experimental work discussed previously, the following hypothesis as been developed to support, direct and validate the research objectives: **A patent-based analogy search tool using functional representations can be used to identify non-obvious functional analogies for design concept generation, and those analogies can be used within the conceptual design process to improve the ideation result as measured by quantity or novelty of ideas.**

SCOPE AND OVERVIEW OF CHAPTERS

This dissertation describes the development of a patent search engine for the express purpose of extracting functional analogies to enhance the conceptual design process. The tool development process is founded on techniques and procedures from the established literature. Chapter 2 presents an overview of field of Information Retrieval and the development of a Boolean-based analogy search tool. Following some initial positive results that illustrated patent analogies can enable novel solutions; several limitations of this search model are identified. The conclusion is reached that a more sophisticated search method is needed to fully implement a functional analogy search

engine that encompasses the entire design space. In Chapter 3, the Vector Space Model search engine development is described. Significant effort is put forth to expand the Functional Basis by extracting functions directly from the patents until a converged, finite set of functions is established. Automated document indexing techniques are employed to create a database of patent functional representations. In Chapter 4, two case studies are presented where the patent functional database is queried using a query generation interface and the resulting analogous patents are used to determine novel solutions to the design problems presented. Chapter 5 presents a controlled experiment designed to evaluate the search engine efficacy during the conceptual ideation process. The two parameters explored are the effect of including analogous patents during brainstorming as well as evaluating the effect of functionality level. The metrics used in the evaluation are the quantity of ideas generated and the average novelty of ideas generated. Finally, Chapter 6 summarizes the results of the search engine development and experimental evaluation as well as discusses future work.

Chapter 2: Patent-based Search Tool Development

INTRODUCTION TO INFORMATION RETRIEVAL & SEARCH ENGINES

A general definition of Information Retrieval (IR) is finding material or documents of an unstructured nature (i.e. text) that meets the requested requirements from within a large collection or database (Manning et al., 2009). Prior to the introduction of computerized databases, IR was an activity engaged in by specialized professionals such as librarians, paralegals and academic researchers. The modern field of computerized IR grew out of key developments in the 1960's and 1970's. Among these significant contributions were the Vector Space Model of document retrieval and the creation of quantitative metrics to evaluate the effectiveness of a search engine algorithm (Salton, 1971, Cleverdon, 1967). With the advent of the World Wide Web, multiple algorithms were developed for searching web-based information sources building on these early innovations, and the most well-known is the commercial Google search engine (Kumar et al, 1999, Brin and Page, 1998). Web browsers utilize web crawling programs to index the World Wide Web and store compressed versions of the web pages in their databases. The original web browsers used simple text search algorithms to generate search results, but the key to the success of Google's search engine was the ability to rank the relevance of a web page hit using variables such as the number of links to the page and the reliability of the linking sites.

The two well established measures for evaluating the performance of an IR system are *precision* and *recall* (van Rijsbergen, 1979). Precision is defined as the percentage of documents retrieved that are relevant to the information requested given in the following equation:

$$Precision = \frac{Relevant\ Docs \cap Retrieved\ Docs}{Retrieved\ Docs}$$

Recall is defined as the percentage of the relevant documents that were actually retrieved given in the following equation:

$$Recall = \frac{Relevant\ Docs \cap Retrieved\ Docs}{Relevant\ Docs}$$

Although the metrics are straightforward, evaluating the performance of a specific algorithm is difficult unless all of the information contained in the document collection is known *a priori*. Additionally, the *relevancy* of a document must be evaluated by subject-matter experts and is subject to variability across evaluators. Until the early 1990's, algorithms could only be tested on small collections and each collection varied from research group to research group. In 1992, NIST in conjunction with DARPA solved this problem by establishing the Text REtrieval Conference (TREC) to define standard test collections. The amount of data in the typical test collection immediately increased by 80 fold, and the rapid, pervasive deployment of IR systems throughout industry and the web can be directly attributed to the efforts of NIST and TREC (Rowe, et al. 2010).

Two of the most widely used IR models to come out of these efforts are the Boolean search methods and the Vector Space Model method (Salton and McGill, 1986, van Rijsbergen, 1979, Manning et al., 2009). Both methods were investigated for implementation of the Patent Analogy Search Engine. The first implementation of the search engine utilized the Boolean approach and is discussed in detail in the following sections.

BOOLEAN MODEL INFORMATION RETRIEVAL

The Boolean model of IR is one of the earliest developed and most widely used models (Manning et al., 2009). This approach is employed in virtually all commercial search engines including Google, LexisNexis, and the USPTO patent search interface. The benefits of the Boolean model are ease of implementation and conceptually intuitive

operation. The query structure takes on the form of nested AND/OR statements constructed from a set of indexed terms derived by parsing the documents contained in the collection. (A detailed description of the parsing and indexing process is provided in Chapter 3).

Once the query is generated, the documents are retrieved by conducting a series of Boolean intersection (AND) and union (OR) operations per the nested structure. The structure of the nesting itself can have a great effect on the recall and precision of the results, where typically AND operations increase precision at the expense of recall and OR operations have the opposite effect (Eastman and Jansen, 2003). One major drawback to Boolean queries is the strict structural requirements of the syntax. Novice users often have difficulty formulating appropriate queries as well as selecting successful terms, although this disadvantage can be overcome with training and experience (Topi and Lucas, 2004).

The objective of the initial version of the patent search engine was to provide a more intuitive and automated interface for generating patent search queries. Previous research has shown automated tools that generate properly structured Boolean queries utilizing a guided process can improve the success of information retrieval (Van Der Pool, 2002). The tool described in the following section was designed to replace the experience and training traditionally required to develop successful queries.

BOOLEAN–BASED ANALOGY SEARCH ENGINE

The initial patent search tool was created to utilize functional modeling representations and terminology, in particular the Functional Basis which was used as the glossary from which search terms were chosen. The Analogy Search Query Generator interface shown in Figure 5 was developed using the MATLAB™ programming

language. See Appendix A for source code. The query generator acquires specific information from the user about the design problem, parses that information, and then constructs a syntactically correct Boolean search query. The formatted query is then sent to a commercial patent search engine to perform the actual patent retrieval. The commercial engine used for this tool was the Freepatentsonline.com search portal (www.freepatentsonline.com). This portal was chosen due to the fact it allowed open access to all digitized patents from 1973 to present, provided rudimentary relevance ranking to retrieved documents, and provided well documented and sophisticated Boolean search capabilities. As an added feature, the patents can be retrieved in either text file format or PDF format that includes all patent illustrations.

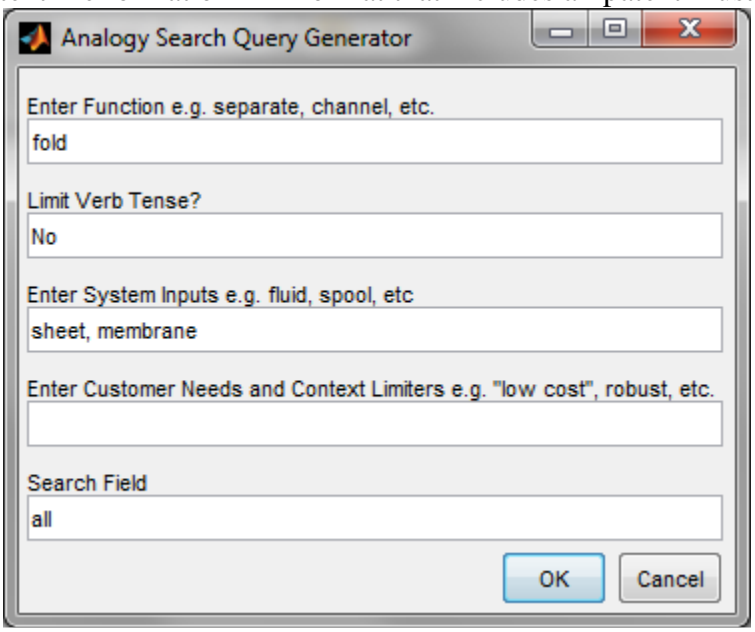


Figure 5: Analogy Search Query Generator user interface

The input fields available to the user are *Functions*, *Limit Verb Tense*, *System Inputs*, *Customer Needs and Context Limiters*, and *Search Field*. The *Functions* field is where the system functions derived from the function structure or functional decomposition are

input. To maintain consistency across users, the Functional Basis functions were used as the term list source.

The Limit Verb Tense field controls the level of function suffix stemming. The default value of *No* means any version of the root function term is included in the search. For example, the basis function *connect* would be mapped to connect, connects, connected, connecting, connector, connection, etc. This is the default behavior of most search engines. Limiting the verb tense by setting the value to *Yes* limits the search to verb and gerund forms of the root function: connect, connects, connecting, and connected. More details on word stemming, its benefits and limitations are discussed in Chapter 3.

The System Inputs field was included to enable the user to input flow domain information in the search query. As defined, the system inputs are the flows into the black box representation of the design problem, for example heat, fluid, force, signals, etc. These flows establish the solution domain of the design problem and can serve to improve the precision of the search results. Unfortunately in terms of analogy search, higher precision also implies only near-field analogies since by definition the patents will be in the same domain. The later iterations of the search methodology omit flow domain information to better enable far-field analogy retrieval which has been shown to improve concept novelty (Cagan et al, 2011).

Customer Needs and Context Limiters were initially added in the search fields to enable information gathered from Quality Function Deployment to be included in the query (Hauser and Clausing, 1988, Otto and Wood, 2001). Experience using the query generation tool found that adding CNs and context limiters or constraints reduced the recall for a particular query, but did not add value to the results. It also has a tendency to

constrain the results to a particular domain similar to the effect of including flow variables. This field was also eliminated in the later version of the search engine.

The final field controlled which section(s) of the patent were searched. The default value of *Abstract* limited the search only the abstract portion of the patent. Other acceptable values were *Description*, *Title*, and *All*. Abstract was the default and most utilized because it enabled quick browsing through multiple patents which was necessary due to the fact Boolean search does not enable robust measures of relevancy. There is no weighting for the relative importance of one search term versus another.

To further enable efficient browsing of search results, a second interface was developed called the Search Viewer. This interface provided multiple layers of information and functionality. One of the primary functions was to group the search results by patent classes. This process of creating coherent groups of related documents is called clustering and enables quick evaluations of large groups of documents (Salton and McGill, 1986, Manning, 2009). The patent class architecture perfectly lends itself to utilization as a clustering category in that by definition a class describes a very specific application or domain that the patent embodies. In Table 2, a selection of primary patent classes is given as illustration.

Table 2: Examples of randomly selected primary patent classes

Primary Patent Class Number	Patent Class Title
42	Firearms
123	Internal Combustion Engines
134	Cleaning and liquid contact with solids
381	Electrical audio signal processing systems and devices

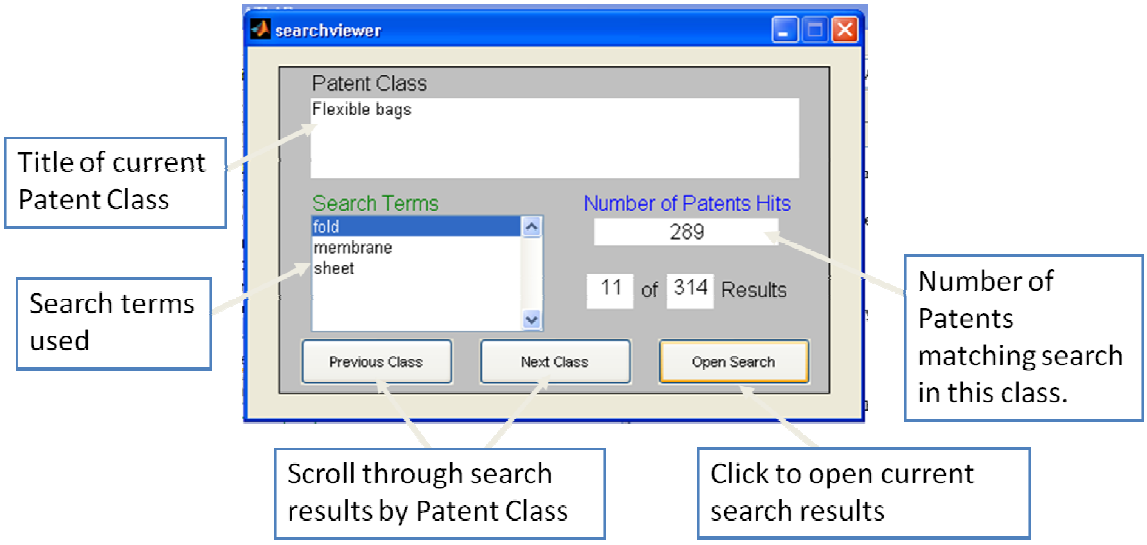
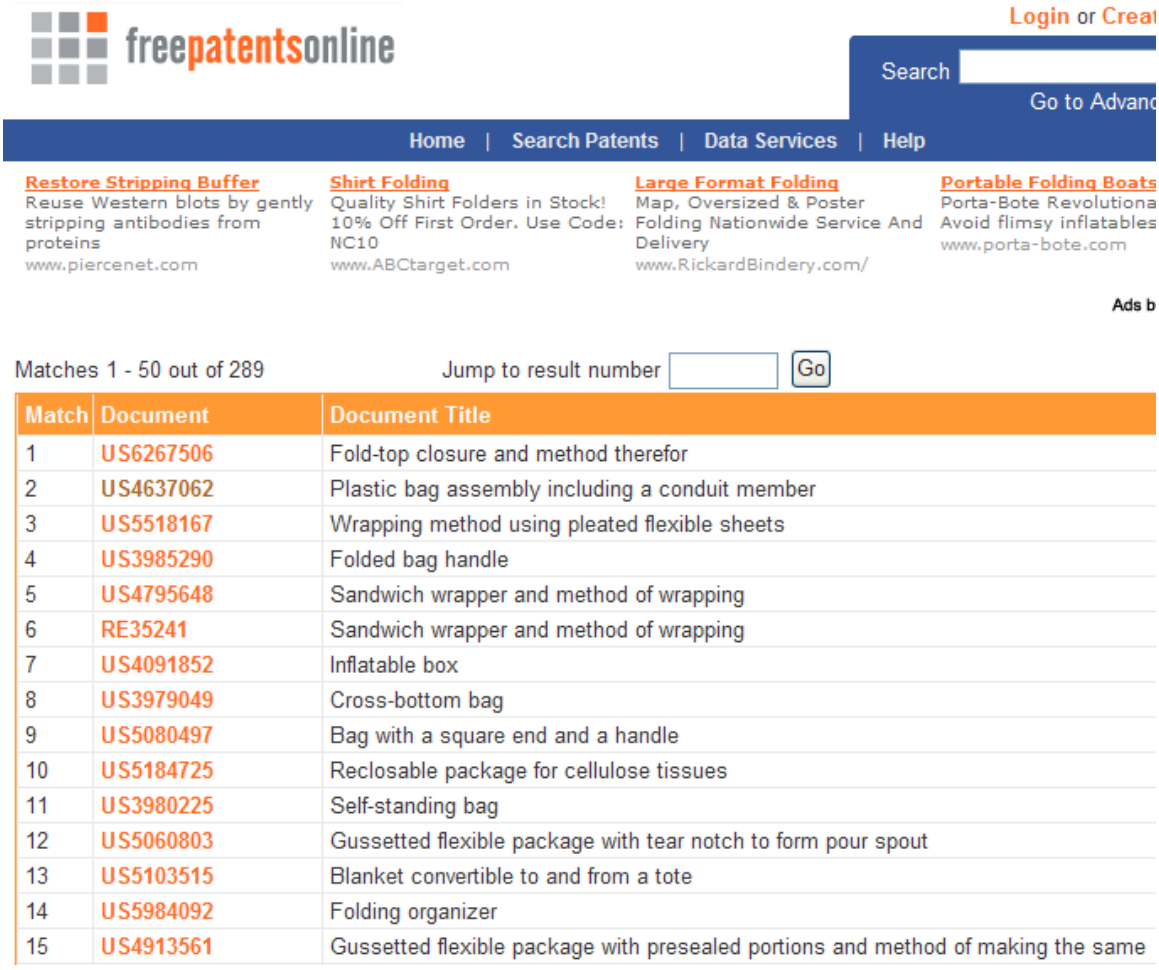


Figure 6: Search Viewer interface for browsing patent search results.

Once the search is completed, the Search Viewer in Figure 6 displays a summary of the search terms used to generate the query as displayed in the *Search Terms* box. The *Patent Class* window displays the title of the current patent class cluster while the *Number of Patent Hits* box displays the number of retrieved patents for the current class. In the example above the *11 of 314 Results* indicates that the current class is 11 out of 314 different classes that return at least one patent. The user can scroll through the classes using the *Previous* and *Next Class* buttons. Finally, once an interesting class is reached the search results are retrieved by selecting the *Open Search* button.

Again, the patent search site Freepatentsonline.com is used as the proxy to retrieve the relevant results. Figure 7 shows the results for the example in Figure 6 above for the query terms: fold (function), sheet (system input), and membrane (system input). The Boolean expression produced by the query generator for this combination of terms is:

(fold OR folds OR folding OR folded) AND (sheet OR sheets OR membrane OR membranes)



Match	Document	Document Title
1	US6267506	Fold-top closure and method therefor
2	US4637062	Plastic bag assembly including a conduit member
3	US5518167	Wrapping method using pleated flexible sheets
4	US3985290	Folded bag handle
5	US4795648	Sandwich wrapper and method of wrapping
6	RE35241	Sandwich wrapper and method of wrapping
7	US4091852	Inflatable box
8	US3979049	Cross-bottom bag
9	US5080497	Bag with a square end and a handle
10	US5184725	Reclosable package for cellulose tissues
11	US3980225	Self-standing bag
12	US5060803	Gussetted flexible package with tear notch to form pour spout
13	US5103515	Blanket convertible to and from a tote
14	US5984092	Folding organizer
15	US4913561	Gussetted flexible package with presealed portions and method of making the same

Figure 7: Search results for fold (function), sheet (system input), and membrane (system input). www.freepatentsonline.com

The individual patents can then be reviewed to extract any analogies where possible. The effectiveness of this analogy search procedure was informally evaluated with three groups of mechanical engineering senior design teams. They were asked to execute patent searches during the concept generation phase. Positive results were reported from utilizing the patent search. For example, in one case, the design problem was related to developing a new system for a robotic manipulator for Los Alamos National Laboratory

(Kottlowski, Halverson and Smith, 2007). Two analogous patents were identified as illustrated in Figure 8.

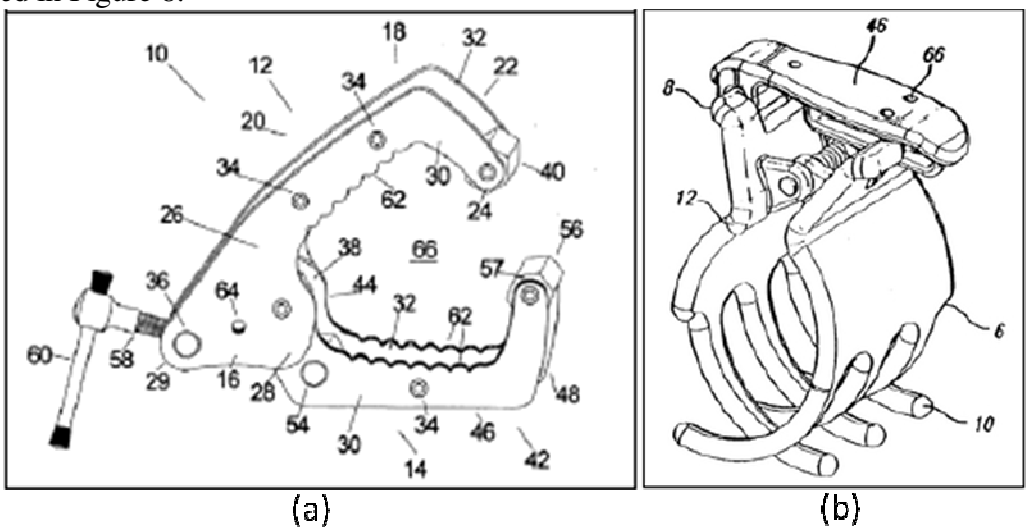


Figure 8: Analogies for robotic gripper found using patent search a) Nutcracker, b) Hair clip

Given the superficial and domain similarity to the design problem, these patents would be classified as near-field analogies to the robotic manipulator design problem. The next step in the search engine development was targeted at implementing features that would bolster the probability of extracting far-field analogies.

One of the primary limitations of the basic Boolean search is a one-to-one mapping between the query terms and document terms which makes the task of identifying far-field analogies more difficult. There is little or no ability to utilize synonymy to capture a range of patents that possess the same general functionality. Functional synonyms in the terminology and structure of the Functional Basis are equivalent to the functional correspondents (Chapter 1, Table 1). In addition, the reconciled Functional Basis is only a small subset of terms used to define functionality in the patent database, i.e., there exist a range of correspondents, synonyms, hypernyms, and

troponyms to the Functional Basis set. In the following chapter, term extraction methods are utilized to derive a complete set of function terms from the USPTO collection. Furthermore, the process and implementation of a prototype Vector Space Model patent analogy search engine is defined and the results of preliminary case studies are presented.

Chapter 3: Vector Space Model Patent Search Engine

VECTOR SPACE MODEL INFORMATION RETRIEVAL

The Vector Space Model (VSM) of information retrieval was first developed in the early 1970's to overcome several limitations of the Boolean model such as lack of search result relevancy ranking, strict query syntax requirements, and query expansion limitations (Salton, 1971, Salton, et al., 1975). In the VSM model, a document is represented as a vector of terms. The terms are words and/or phrases extracted from the documents themselves using natural language processing techniques (van Rijsbergen, 1979, Rindflesch, 1996). The collection of terms fall into one of two categories: controlled vocabularies and uncontrolled vocabularies. Uncontrolled vocabularies use automated term indexing processes to extract all significant terms without consideration for the informational content. Controlled vocabularies utilize extraction processes to build in specific concepts and information into the vocabulary. One such example is the biomedical controlled vocabulary of Rindflesch and Aronson (2002) created to enhance information retrieval from within medical databases. The Functional Basis (Hirtz, et al, 2002) is another example of a controlled vocabulary where the terms have specific contextual and conceptual meanings, i.e. function and flow, which can be used to retrieve patents based on functional information.

To represent a document as a vector of terms, each term in the vocabulary becomes an independent dimension in an *n*-dimensional space where *n* is the number of vocabulary terms. All of the documents in the database are mapped onto the vector space using indexing algorithms. In the most basic algorithm, binary values are assigned for each dimension according to whether the term occurs in the document, 1 for present and 0 for absent, but typically a weighting factor is applied to the occurring terms (Salton and McGill, 1986). The two common weighting factors are the *term frequency* (*tf*) which is

the frequency of occurrence within a specific document and the *document frequency* (*df*) which is the frequency of occurrence across documents (Manning, et al., 2009, Salton and Waldstein, 1978). The specifics of the term weight derivation and usage within the context of this research are described in detail in the patent indexing section of this chapter.

The resulting term-document matrix is a matrix of size *m* x *n*, where *m* is the number of documents in the collection and *n* is the number of terms, that is typically a very sparse matrix given that relatively few terms occur within a single document. A variant of the standard VSM model called Latent Semantic Indexing (LSI) or Latent Semantic Analysis (LSA) can be used to reduce the dimensionality of the term-document matrix (Manning, et al, 2009). Using term co-occurrence information, Singular Value Decomposition (SVD) methods map the document terms to a reduced *concept* space (Moldovan, et al., 2005). In this context, *concepts* are groups of terms that are synonyms, hypernyms, and troponyms of each other. For example, the terms *car, truck, pickup* and *automobile* are synonyms and/or hypernyms, where a hypernym is defined as a generalized term that more specific terms fall under. Troponyms apply only to verbs and are defined as verbs that more specifically describes the action. For example *march* is a troponym of *walk*. Using SVD, the four terms can be clustered into a single dimension. Applied across the entire term-vector, the *n*-dimensional space typically in the thousands of terms is reduced to a *k*-dimensional space typically in the hundreds of concepts and the dimensionality of *k* is a system parameter that must be tuned to optimize the mapping (Dumais, 1995).

Some drawbacks to LSI are high computational requirements for the SVD algorithm and difficulties in adding documents to the database. Adding large numbers of new documents to a database without recomputing the SVD can lead to skewed similarity

results and omitted terms. This issue is particularly significant in the patent database where documents are continually added (Manning, et al., 2009). Conflicting reports of the performance improvement relative to the standard VSM query are reported in the literature. Dumais (1995) reports an average performance increase of 5%, and Moldovan, *et al.* (2005) found only a 5% improvement over VSM for application of LSI specifically to patent searches. Given the added computational overhead, issues with document additions, and marginal performance improvement, the standard VSM approach was chosen over LSI as the search engine model for this research. Issues of polysemy, where a word has multiple meanings, and synonymity, where multiple words have the same meaning, are overcome through query mapping heuristics using one-to-many term mapping.

One of the powerful aspects of the VSM model is queries can be mapped to the term vector space using the same algorithms as the document mapping. This flexibility removes the syntactic constraints on the query structure and provides a simple, straight-forward metric for evaluating similarity between the query and the documents (Manning, et al., 2009). In the term-vector space, the similarity between the query vector and a document vector is equivalent to the angle between the vectors. In Figure 9, the cosine of the angle between the vectors is a commonly used metric since it has the useful properties of varying from 0 for orthogonal vectors and 1 for identical vectors (Salton et al, 1975).

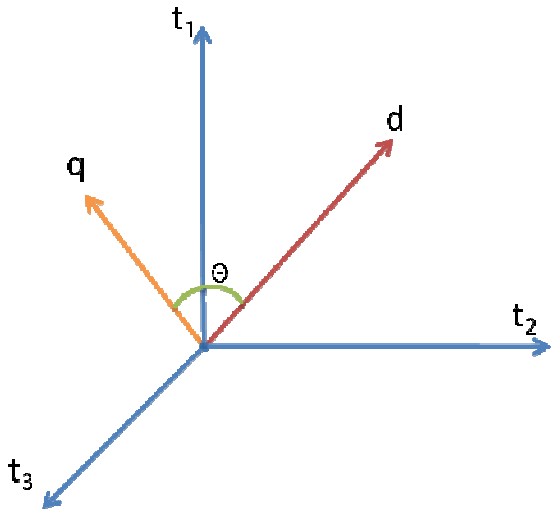


Figure 9: Example of document, *d*, and query, *q*, vectors in a three (3) term vector space (*t1*, *t2*, *t3*). Similarity between vectors is the cosine of angle ϴ.

Conveniently, a nearly identical metric was used by McAdams and Wood (2002) to measure the functional similarity between functional models (McAdams, et al., 1999, Stone, et al., 2000c).

The final aspect of the VSM model which was exploited in this research was the capability to establishment query mapping rules such that a single query term is mapped to multiple document terms. The ability to utilize this synonymy allows a simplified query to capture a range of patents that possess the same general functionality. The synonymy mapping process as well as the function-based controlled vocabulary construction, patent document indexing, and weighted similarity metrics are discussed in detail in the follow sections.

VECTOR SPACE MODEL-BASED ANALOGY SEARCH ENGINE

The development and implementation of the VSM analogy search engine is a five step process shown in Figure 10 which begins with constructing a controlled vocabulary

of functions extracted from the patent database, building on the hierarchical structure of the Functional Basis. Once a complete set of function terms is compiled, the patent documents are indexed against the expanded functional basis to create a vector representation of the patent database.

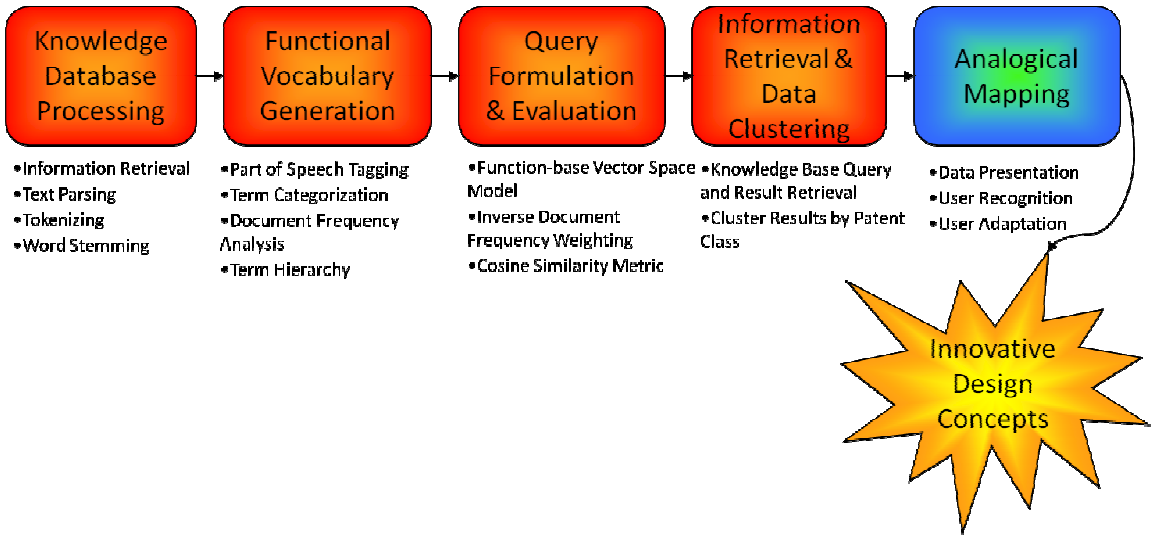


Figure 10: Vector Space Model search engine development and implementation methodology

Query generation and similarity ranking tools are then developed to query and retrieve the patents with the highest degree of relevance to the functional description of a given design problem. Finally, the most relevant patent results are presented to the user. In Chapter 4, two case studies are presented to verify and validate the efficacy of the search tool for identifying analogous patents which are used to create novel design solutions.

Controlled Vocabulary of Functions Construction Methodology

One of the primary goals of this research is to identify and extract a *complete* set of functions that covers the entirety of the patent database. The completeness of the function controlled vocabulary is evaluated using two metrics: 1) cumulative unique functions versus number of patents indexed and 2) function-document frequency versus

order terms are identified. The convergence criteria are discussed in detail in the next section, but before convergence can be evaluated, the terms must first be extracted.

Manual Indexing of Patents

Document indexing is the process of transforming the complete text of a document into a limited set of identifiers or terms and both manual and automated indexing procedures exist in the literature (van der Meulen, and Janssen, 1977, van Rijsbergen, C. J., 1979, Manning et al, 2009). The initial approach used in this research was a manual indexing procedure, where a subject-matter expert extracts the terms according to the desired information. The subject-matter expert, in this case, is well versed in functional modeling, representations, and vocabularies. A preliminary set of 1000 patents were manually indexed. The patents were chosen using a random patent number generator utility created specifically for this research. The indexing process was decidedly low-tech; hardcopies of the patents were physically marked up and the data was transcribed into an Excel spreadsheet. This indexing process is very accurate, but extremely tedious and time consuming. Indexing enough patents to reach a converged, complete set of functions is not feasible. The manual indexing process is a useful exercise to develop deeper understanding of the details and structure of patents and particularly how function is represented in patent documents that are produced to specifically describe form. The primary insight elucidated from the manual indexing process is that functional terms are frequently in the form of present participles and gerunds, in addition to action verbs. Present participle forms, where the verbal is used as an adjective, are very common and second only to action verbs in frequency. The abstract from the patent for a connecting rod (Wandel, 1981) is a representative example of this structure:

> A *connecting* rod assembly for a two-stroke cycle *reciprocating* piston internal combustion engine has the piston pin axis slightly *skewed* from the crank-shaft axis to provide favorable conditions for *lubricating* the piston pin.

where the participles *connecting* and *reciprocating* are indexed as the functions *connect* and *reciprocate*. The gerund *skewed* is indexed as *skew*, and lastly, the verb *lubricating* is indexed as *lubricate*.

Once the structure and forms of functions within patents are better understood, tools can then be developed to improve the efficiency of the indexing process. In the next embodiment of the indexing procedure, the term classification continues to be conducted manually, but patents are preprocessed to distill the contents down to only the significant terms. The first step in the process is using regular expression algorithms (MATLAB, 2009) to manipulate and parse the patent information such as class, number, and to individually extract specific sections of the patent text such as abstract, title, and description directly from the HTML text obtained from the USPTO or other web-based patent databases. Figure 11 shows the HTML source code for the Connecting Rod patent, and the embedded abstract text.

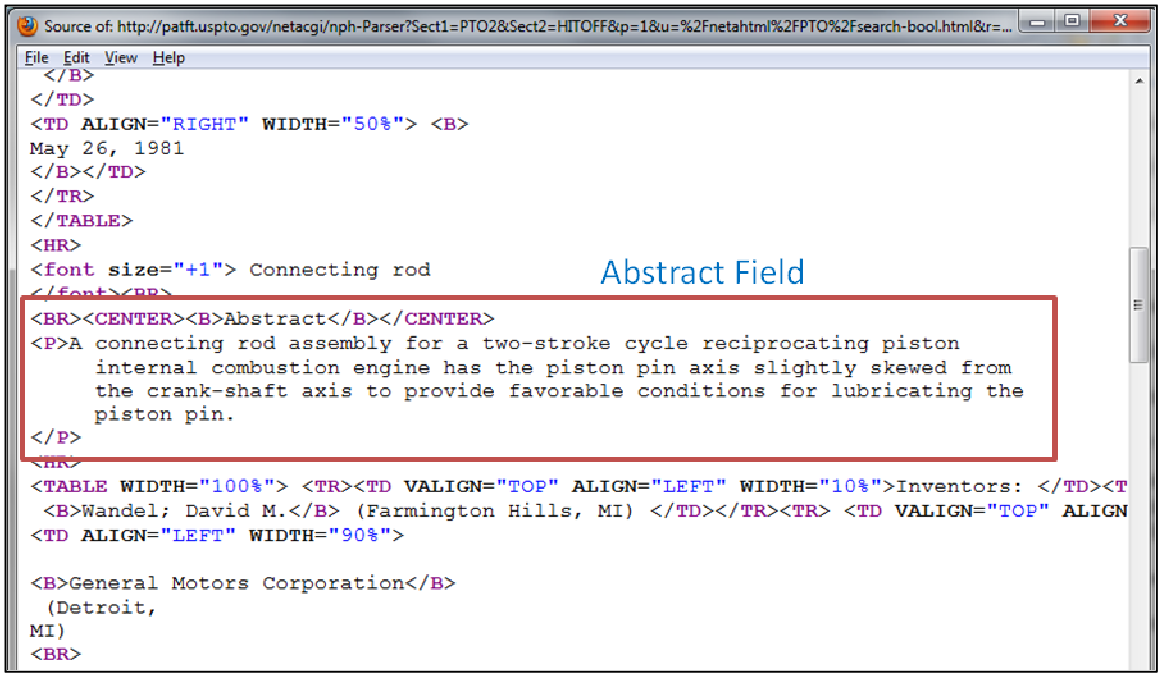


Figure 11: HTML source for abstract text of connecting rod patent 4,269,083 (http://patft.uspto.gov/netahtml/PTO/search-bool.html, Wandel, 1981)

Regular expression functions are formal text manipulation functions built on text pattern matching. Using pattern matching with a structured HTML tagging scheme enables the desired information to be easily parsed from the HTML source code. Custom parsing schemes for both the USPTO and Freepatentsonline.com were created for this research and an example of the code used to extract abstract text from Freepatentsonline.com is given in Figure 12.

```
[m s e] = regexp(url, '(Abstract:</div>)\s+\S+\s+\S+\s+', 'match','start', 'end');
if isempty(m)
   indx_terms = ' ';
   indx_words = indexed_words;
   return
end
[m2 s2 e2] = regexp(url, '\s+(</div>)', 'match','start', 'end');
```

Figure 12: Regular expression function for parsing abstracts from Freepatentsonline.com

Once the text is parsed from the online databases, it is further processed to remove superfluous terms such as prepositions or excessively common terms using a *stop word list* (Salton, G. and McGill, M.J., 1986). An excerpt from the complete stop word list is given in Table 3. All punctuation is also removed in this step. Custom regular expression functions are used with character matching to remove the stop words and punctuation. The complete parsing code is contained in Appendix B.

Table 3: Excerpt from Salton's stop word list (Salton, G. and McGill, M.J., 1986).

a	although	anywhere	behind	cant
able	always	are	being	co
about	am	around	below	con
above	among	as	beside	could
across	amongst	at	besides	couldnt
after	amoungst	back	between	cry
afterwards	amount	be	beyond	de
again	an	became	bill	describe
against	and	because	both	detail
all	another	become	bottom	do
almost	any	becomes	but	done
alone	anyhow	becoming	by	down
along	anyone	been	call	due
already	anything	before	can	during
also	anyway	beforehand	cannot	each

Removing these non-significant terms can reduce the size of a document by between 30 and 50 percent, greatly reducing the indexing effort (van Rijsbergen, C. J., 1979). Following parsing, the connecting rod patent abstract (Wandel, 1981) becomes a simplified string of terms given as:

> connecting rod assembly two stroke cycle reciprocating piston internal combustion engine piston pin axis slightly skewed crank shaft axis provide favorable conditions lubricating piston pin

After parsing is completed, word stemming algorithms are applied to the text to further consolidate terms and to extract the *root function* terms. Suffix stripping is a common word stemming process, and initially a standard algorithm reported by Porter (1980) is used for this purpose. The Porter stemmer is very efficient but is determined to be too aggressive for the purpose of this research due to the fact the algorithm strips suffixes that contain part of speech content, obfuscating the distinction between nouns and verbs. An example is both component-noun terms *connector* and *connection* are stemmed to the term *connect* which is a function-verb, resulting in an inflated document frequency for the function *connect*. The intent of this research is to only consider verbal forms of function and not component-nouns with a possible verbal root due to the analogy mapping advantages discussed in Chapter 1. The prefix stemming issues were resolved by refactoring the Porter stemmer into a customized 'light' stemming algorithm designed to remove tense and plurality while maintaining part-of-speech intent. The code for the light stemming algorithm is located in Appendix B. Based on patterns derived from the initial round of indexing, a prefix stripping algorithm was created to extract *root functions*. The prefixes identified for removal are given as:

- 'sub','re','un','de','under','mis','over','pre','post','non','counter','out','inter','micro','up','super','en','co','dis','hyper','ultra', 'anti'

Figure 13 illustrates the stemming procedure for a family of verbal forms of the function *connect.*

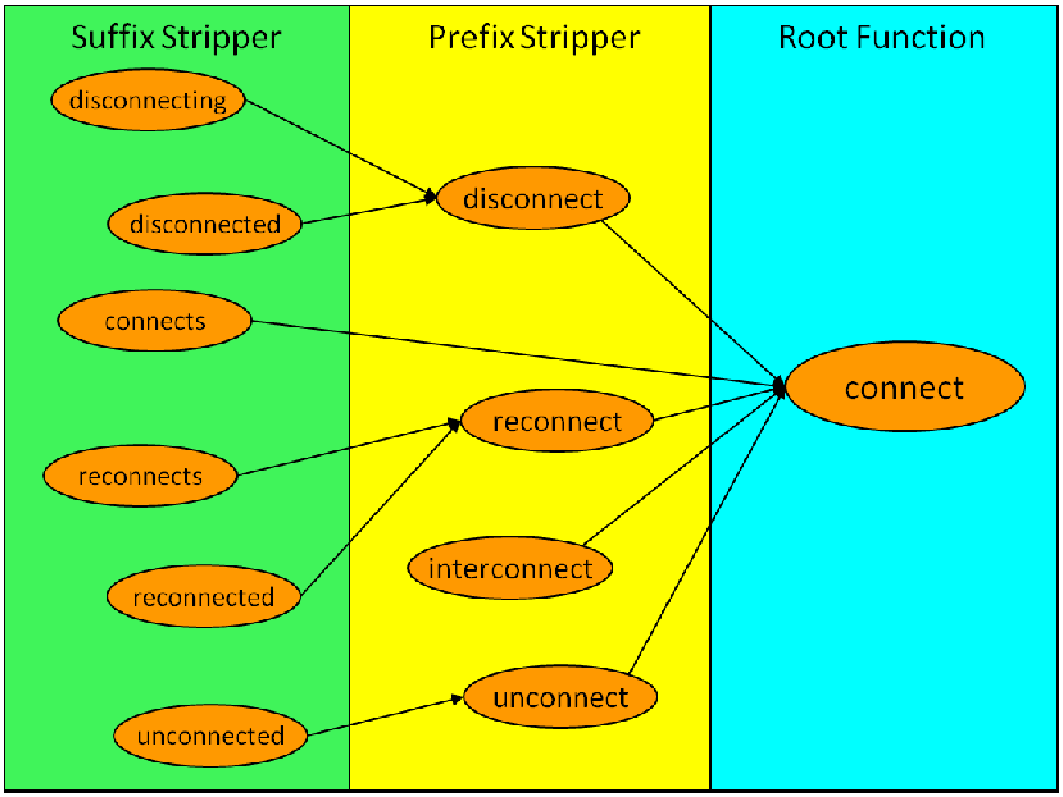


Figure 13: Word stemming with both suffix and prefix stripping algorithms

The code for the prefix stripping algorithm is included in Appendix B. As part of the prefix stemming function, redundant root terms are eliminated as well. Following word stemming, the connecting rod patent abstract (Wandel, 1981) becomes an even simpler string of term given as:

> connect rod assembly two stroke cycle reciprocate piston internal combustion engine pin axis slightly skew crank shaft provide favorable condition lubricate

and the document terms have been reduced by 40% (35 terms reduced to 21).

After parsing and stemming, the surviving terms are sorted into categories derived from the functional modeling lexicon: functions, flows, components and attributes. Functions and flows have been defined previously, (Otto and Wood, 2001, Hirtz et al, 2002). Components are simply physical objects and attributes are terms that describe the

physical appearance, size, geometry, etc. of the components. During this phase of the indexing, all terms are categorized for archiving purposes, but only the functions are used in the search engine development. Usage of components and attributes to augment the patent search are outside the scope of this research and will be left for future work. A prototype Graphical User Interface (GUI) was developed to reduce term sorting time as, shown in Figure 14. See Appendix B for the GUI code.

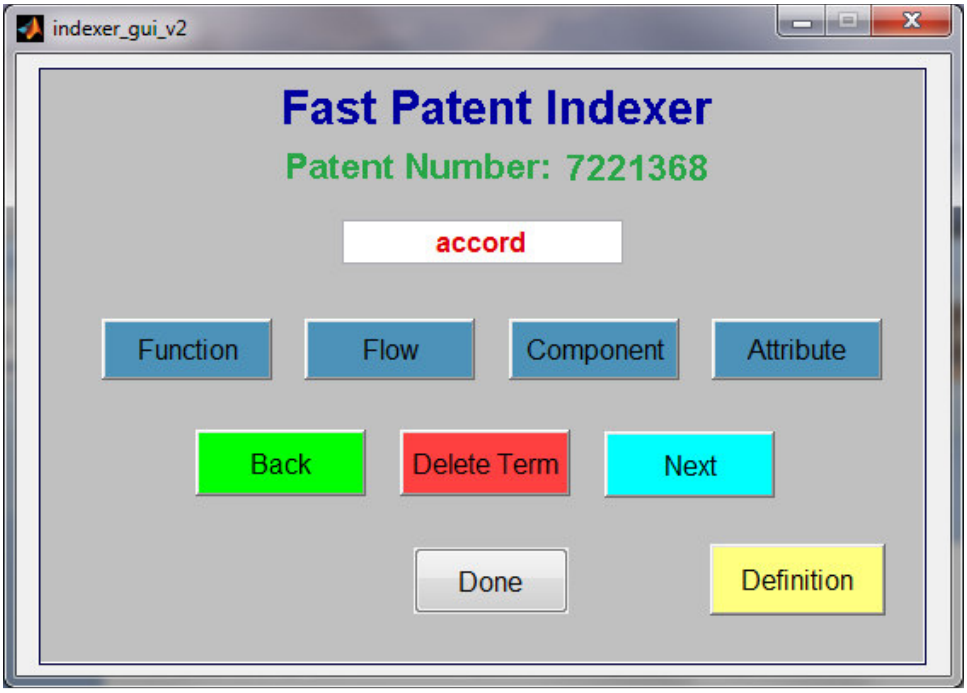
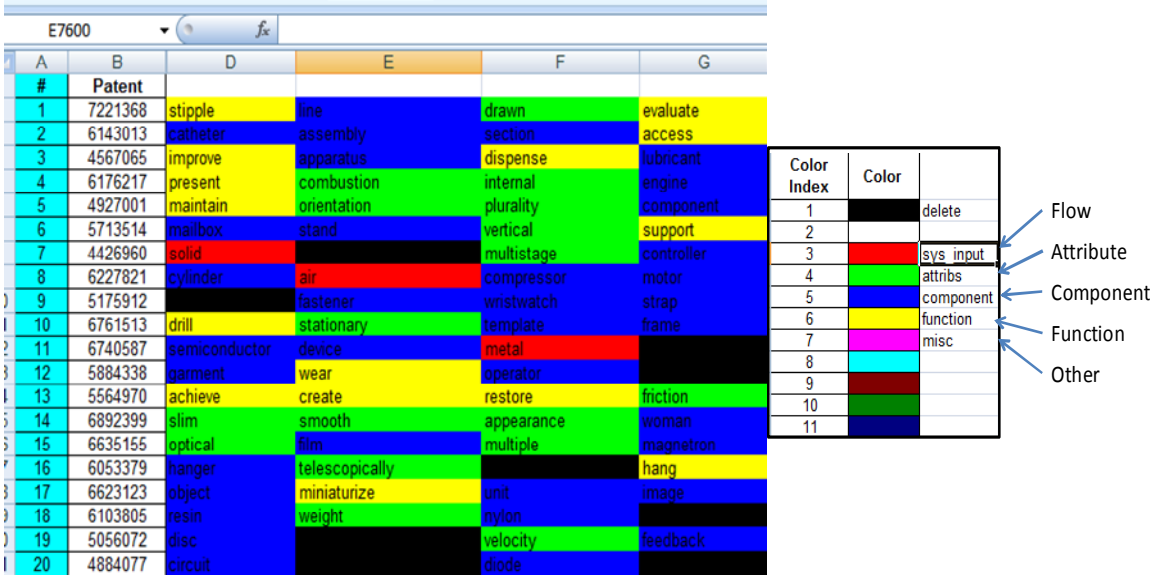


Figure 14: FastPatentIndexer user interface for augmented manual indexing

The patent terms are presented one-at-a-time to the user, and the category is assigned by selecting the corresponding button. The category assignment is chosen based on the category definitions discussed previously by the subject-matter experts. The delete button is used to remove additional non-significant terms or nonsensical terms that survive parsing and stemming and is most often used while indexing chemical patents because

the chemical formulas do not remain intact through the parsing process. The sorted terms are then written to an Excel spreadsheet serving as the term database. A color-coding scheme is implemented for quick identification of a term's category by both the human user and the storage/retrieval functions. A portion of the term database is shown in Figure 15 along with the color-coding scheme. (Note: The "other" category was removed from the final prototype as it adds very little relevant information).



#	Patent	D	E	F	G
1	7221368	stipple	line	drawn	evaluate
2	6143013	catheter	assembly	section	access
3	4567065	improve	apparatus	dispense	lubricant
4	6176217	present	combustion	internal	engine
5	4927001	maintain	orientation	plurality	component
6	5713514	mailbox	stand	vertical	support
7	4426960	solid		multistage	controller
8	6227821	cylinder	air	compressor	motor
9	5175912		fastener	wristwatch	strap
10	6761513	drill	stationary	template	frame
11	6740587	semiconductor	device	metal	
12	5884338	garment	wear	operator	
13	5564970	achieve	create	restore	friction
14	6892399	slim	smooth	appearance	woman
15	6635155	optical	film	multiple	magnetron
16	6053379	hanger	telescopically		hang
17	6623123	object	miniaturize	unit	image
18	6103805	resin	weight	nylon	
19	5056072	disc		velocity	feedback
20	4884077	circuit		diode	

Figure 15: Patent term index database with color-coded term context

The tool-enhanced indexing process was applied to an additional 9,000 randomly chosen patents, using the same random patent number generator, and the initial 1,000 patents were re-indexed using this process to verify the accuracy. All functions extracted using initial manual indexing were all captured using the new procedure with no omissions which is a strong positive confirmation of the procedure's accuracy. Once an index of 10,000 patents was reached, the completeness of the functional vocabulary was evaluated using a cumulative functions versus number of patents indexed metric. The slope of this metric is the rate of new functions added for each additional patent indexed, therefore

once a complete set of functions is compiled, the slope will approach a horizontal asymptote which is the total number of unique functions. The plot for the first 10,000 patents is given in Figure 16.

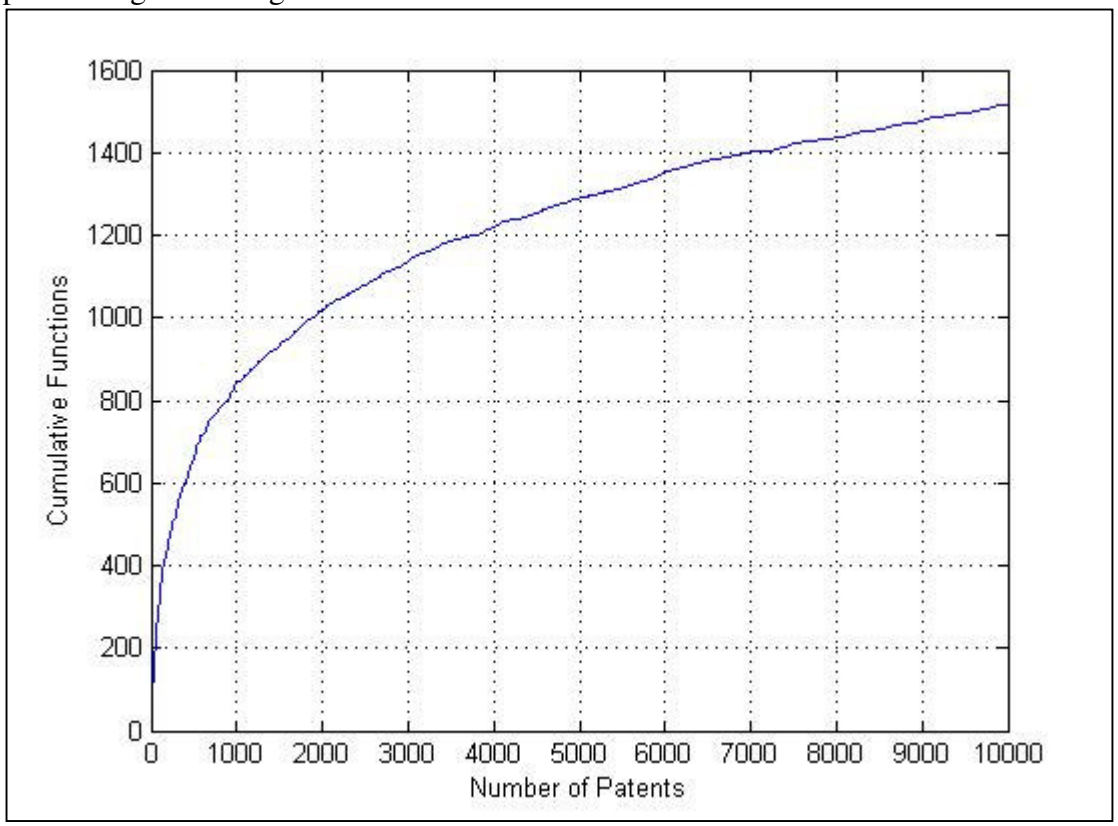


Figure 16: Cumulative functions versus number of patents indexed with positive slope verifying non-convergence of function vocabulary

It can be seen that the curve has a positive slope which is positive confirmation convergence of the function set has not been reached. In fact, the cumulative functions appears to be increasing at a nearly steady rate. In conclusion, additional patents need to be added to the index to reach completeness for the function vocabulary. Even with the additional tools, the manual indexing process is laborious. The next evolution in the

indexing process is implementing an automated process to produce a first-pass category assignment to extracted terms followed by a manual validation step.

Automated Indexing with Part-of Speech Tagging and Manual Validation

The key to automating the indexing algorithm was including a part-of-speech (POS) tagging program into the parsing process. TreeTagger, an open-source POS tagging program, was chosen based on high accuracy of tagging in natural language documents and ease of implementation into the existing indexing scheme (Schmid, 1994). The tagger identifies the POS from sentence structure using probabilistic, binary decision trees. The tagger uses a pre-programmed lexicon of terms with associated POS probabilities that have been extracted from natural language texts such as newspapers and journals. Test of accuracy using the TreeTagger have shown it to be over 95% accurate (Schmid, 1994). The manual verification step is included to validate the POS tagging accuracy within patent documents for verbs, participles and gerunds. Noun and other POS tags were not verified since these terms are not critical to the function vocabulary construction. Tagging errors (nouns tagged as verbs, etc.) were identified during the validation step at a rate of approximately 4% which is comparable to the published data. An example of the tagger output is given in Figure 17.

Original Term	Part of Speech	Stemmed Term
description	NN	description
:	:	:
BACKGROUND	NN	background
OF	IN	of
THE	DT	the
INVENTION	NP	INVENTION
1.	CD	@ord@
Field	NN	field
of	IN	of
the	DT	the
Invention	NN	invention
The	DT	the
present	JJ	present
invention	NN	invention
relates	VVZ	relate
to	TO	to
an	DT	an
adjustable	JJ	adjustable
bed	NN	bed
canopy	NN	canopy
and	CC	and
curtain	NN	curtain
system	NN	system
and	CC	and
more	JJR	more
particularly	RB	particularly
pertains	VVZ	pertain
to	TO	to
providing	VVG	provide

Figure 17: Partial TreeTagger output for the description field of a patent

The terms in the first column are the first sentence of the patent description. In the second column, the part-of-speech tags identified by the TreeTagger program are listed for each corresponding term. For function verb indexing, only the verb tagged terms are extracted and the complete list of verbal tags is given in Table 4. A complete list of tags and descriptions for the TreeTagger software is given in Appendix C.

Table 4: POS tags utilized to extract function verbs

Tag	Description	Example
VV	verb, base form	collect
VVD	verb, past tense	collected
VVG	verb, gerund/present participle	collecting
VVN	verb, past participle	collected
VVP	verb, sing. present, non-3d	collect
VVZ	verb, 3rd person sing. present	collects

With this automated process in place and verified, an additional 55,000 random patents are indexed for a total of 65,000 indexed patents. The number of patents that are indexed (65,000) is limited by the maximum database size constraints. The extracted and tagged terms are added to the existing database and the convergence metrics of both cumulative functions versus number of patents and term document-frequency versus order found are evaluated to check for completeness. A secondary database could be constructed to expand the capability beyond 65,000 patents if completeness has not been achieved, but this step is not necessary per the results presented in the following section.

Function Vocabulary Completeness, Convergence and Conditioning

Once the 65,000 patents are indexed, a set of approximately 1700 functions are identified. In Figure 18, cumulative functions plotted versus patents illustrates that the metric has reached a horizontal asymptote, and furthermore convergence was reached at approximately 61,000 patents.

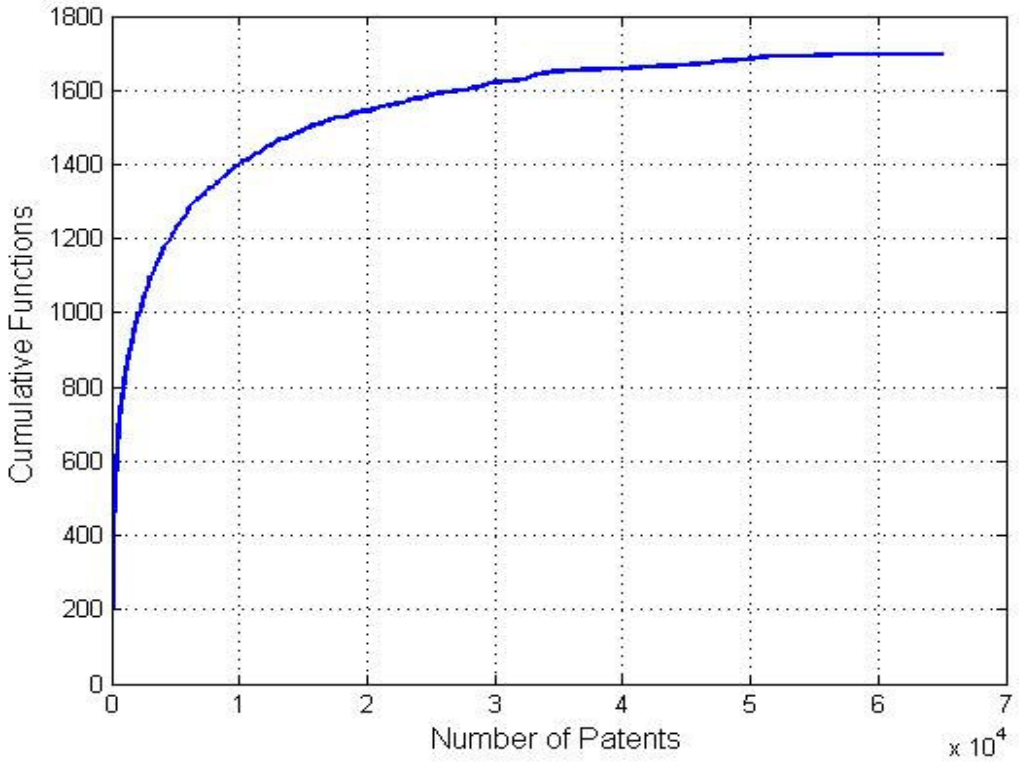


Figure 18: Cumulative functions versus number of patents indexed with horizontal asymptote at ~1700 functions and 61,000 patents verifying convergence of function vocabulary.

A secondary metric for convergence investigated in this research is the document frequency of the function versus the order in with they were identified. This plot is given in Figure 19. The document frequency measures how often a term occurs across all patents. Statistically, high document-frequency terms will be found earlier due to the random sampling.

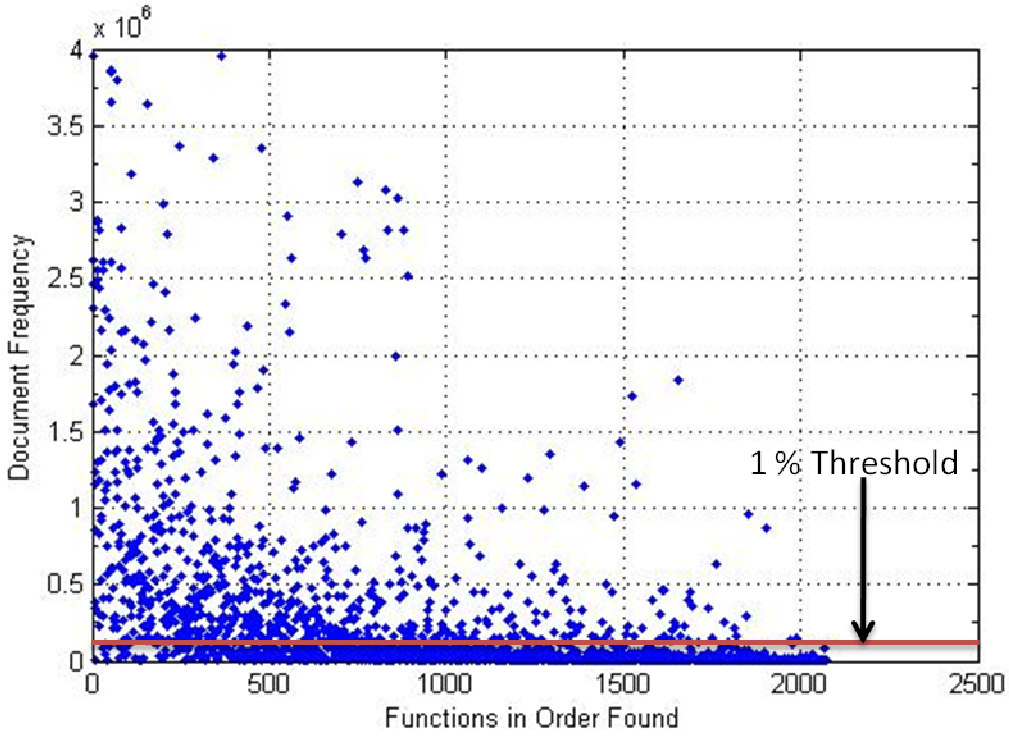


Figure 19: Term document frequency versus order found showing the frequency falls below a 1% threshold (occur in < ~45000 patents).

The trend shown in Figure 19 is clearly confirmed with the functions' document frequencies clustering below 1% of searchable patents as a function of order found. The 1% threshold is chosen not based on a hard limit found in literature, but from the insight that terms below that level are excluded from 99% of the remaining patents. The low resolving power of these low frequency terms means little value is added to search queries by including them, since they will have no impact on similarity for the vast majority of patents. The resolution power of terms as a function of frequency is demonstrated by Zipf's Law (Zipf, 1949, van Rijsbergen, 1979, Wyllys, 1981). In Figure 20, the frequency of words versus the rank order of words (highest frequency to lowest frequency) is shown as well as the resolving power of significant terms.

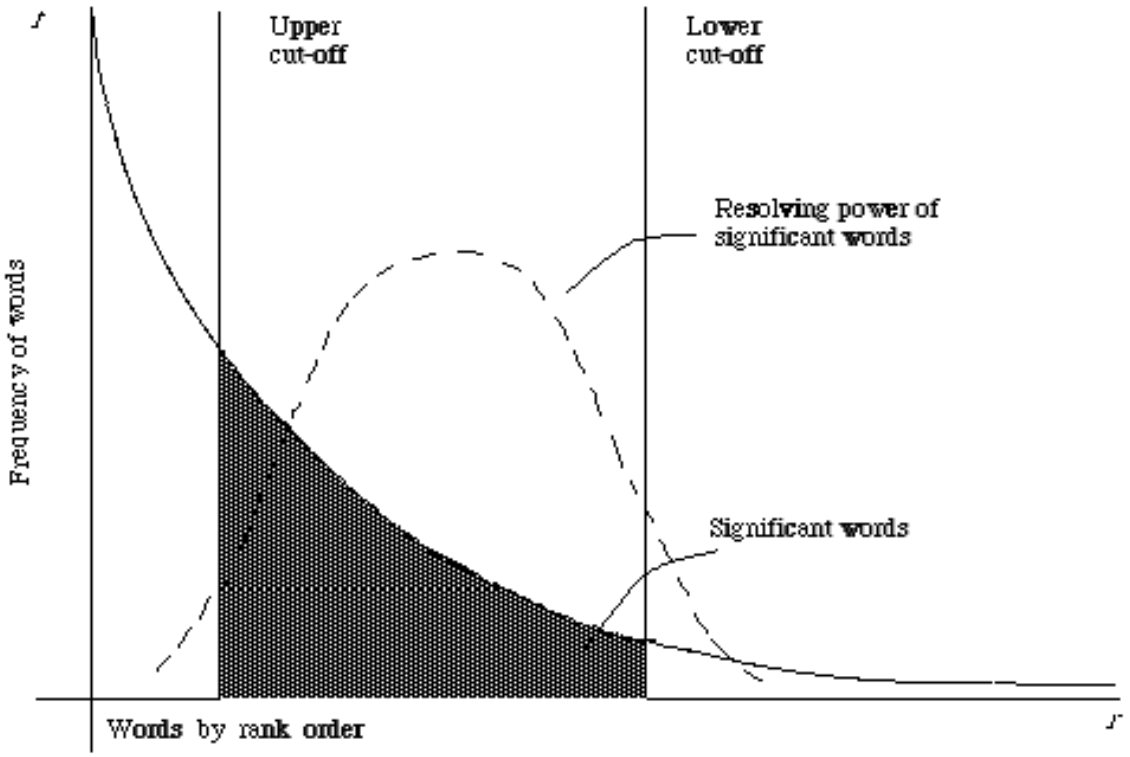


Figure 20: Zipf's Law and the relationship to the resolving power of significant terms. (van Rijsbergen, 1979).

Interestingly, the frequency of words follows a power law distribution straight line on log-log scale) and the resolving power is analogous to a Gaussian distribution which states both very high frequency terms and low frequency terms have low resolving power. This reasoning for the high frequency terms is the theoretical justification for using the stop word lists. The upper and lower cut-offs are theoretical thresholds and no metric was found for determining these boundaries in the literature other than a trial and error tuning process (Manning et al., 2009).

The function vocabulary identified in the indexing process is plotted in Figure 21, using log-log axes. A theoretical Zipf's distribution was fit through the data for comparative purposes.

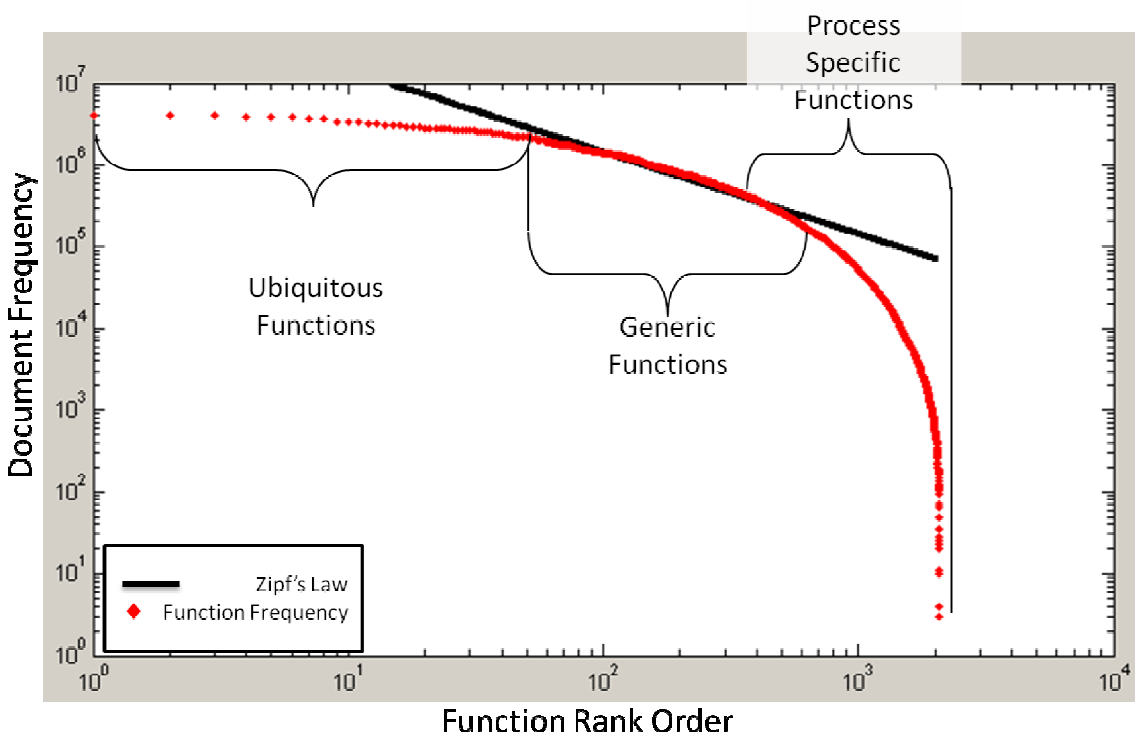


Figure 21: Function vocabulary document-frequency versus rank order comparison with Zipf's power law distribution.

When compared to Zipf's law, three different regimes of function frequency distribution can be identified and are label as: ubiquitous, generic and process-specific. Ubiquitous functions occur so frequently across all patents that they offer little value for determining similarity or relevance, per Zipf's Law theory. These functions can be considered to lie above the upper cut-off chosen to be all terms that occur in more than 50% of patents. Examples of these functions are *provide, use*, etc. The ubiquitous functions, which account for 50 of the 1700 terms, are to be removed from the final function vocabulary index. The complete list of ubiquitous terms is given in Table 5.

Table 5: Ubiquitous functions removed from Functional Basis vocabulary

Ubiquitous Functions		
describe	position	lower
use	line	point
provide	require	allow
form	process	operate
present	illustrate	indicate
include	order	extend
relate	surface	prevent
follow	case	turn
part	determine	obtain
view	connect	improve
end	set	receive
show	increase	condition
time	reduce	select
type	make	separate
close	number	shape
move	produce	space
place	define	size
control	change	
portion	contain	

Generic functions have a good balance between frequency and specificity to enable better distinction between patent vectors within the cosine similarity metric. Examples of these functions are *shape, rotate*, etc. Process-specific functions occur in very few patents and would be below the lower cut-off region in Figure 20. Blindly following the resolving power hypothesis, these terms should be removed from the function index as well, but the rarity of the function may in and of itself lead to novel solutions. The retentions of these few extra terms does not impact the computational overhead since the converged and complete functional vocabulary consists of just over 1700 terms after removal of the ubiquitous functions. The VSM search engine can now

be developed using the functional vocabulary derived in this section of work. The implementation in a prototype search interface is described in the following section.

PROTOTYPE SEARCH ENGINE IMPLEMENTATION

Expanded Functional Basis Development

After the final set of functions is vetted per the process described previously, affinity diagramming and thesaurus construction techniques are used to create a hierarchical structure for the functional vocabulary modeled after the functional basis (Otto and Wood, 2001, Hirtz, et al., 2002). The affinity diagram technique is used to group like-terms together into sub-groups of hypernyms and synonyms. Unusual or unfamiliar words were checked against existing thesauri to select the proper grouping. The iterative process creates Secondary functions with similar numbers of Correspondent sub-functions. The function sub-groups are split or merged accordingly to attain consistent numbers of functions in each sub-group. The detailed procedure for developing the hierarchical structure of the expanded functional basis is given as follows:

1. Sort all terms into Primary basis functions using thesaurus and Wordnet according to synonymy and hypernym relationships.
2. Rank verbs within each Primary group by document frequency
3. Review verbs and extract five highest frequency terms. These terms become initial Secondary functions.
4. Group remaining Correspondent functions within each Secondary group using thesaurus and Wordnet hierarchical relationships.
5. Rank verbs within each Secondary group by document frequency
6. Separate groups which contain more than 50 verbs into multiple Secondary function groups.

7. Iterate on grouping process to produce Secondary function groups with similar number of Correspondent functions.

In Figure 22, the results from steps 1 and 2 are shown with the eight Primary function groups ranked by document frequency according to the frequency of occurrence in the 65,000 patents indexed

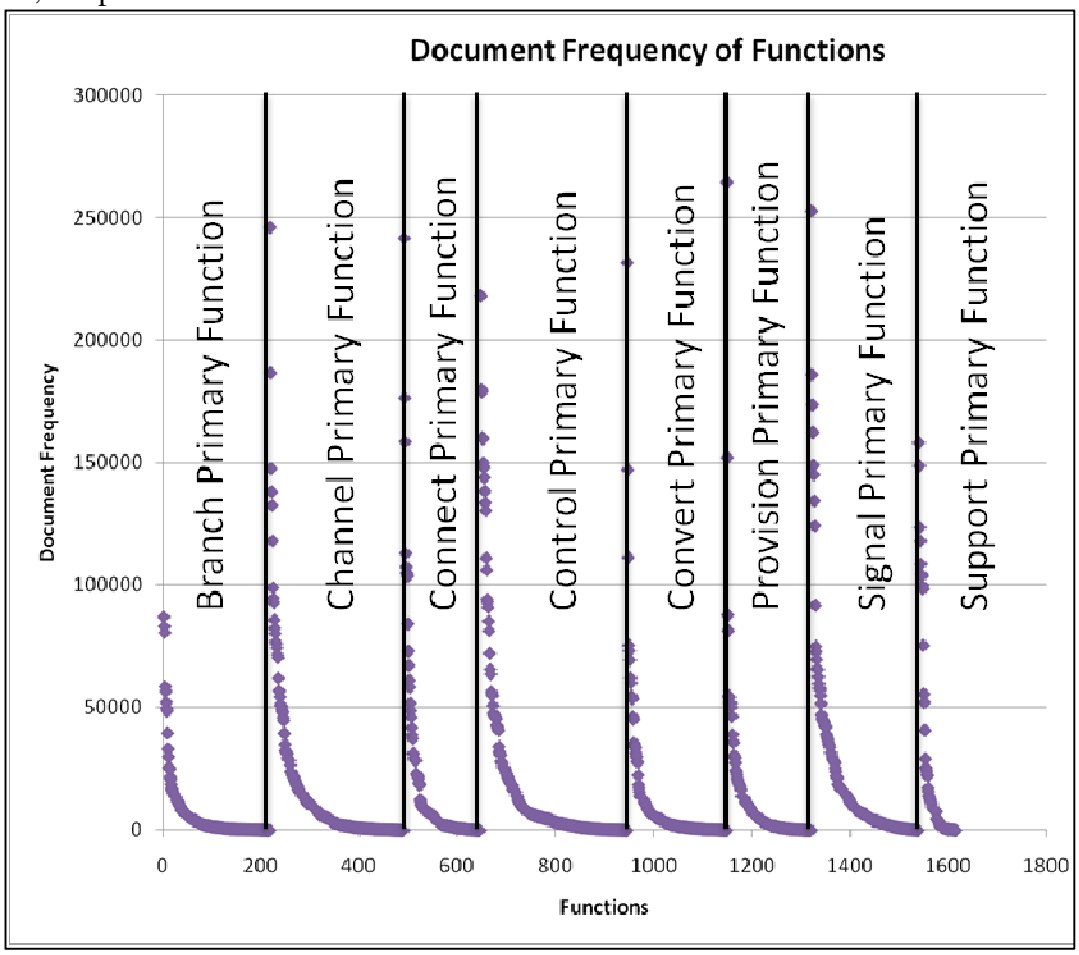


Figure 22: Primary function groups ranked according to document frequency

The same Primary functions from the Functional Basis (Otto and Wood, 2001, Hirtz, et al., 2002) are used as the top-level functions. Secondary level is more highly granularized than the current Functional Basis. Between seven and thirteen Secondary

functions are ranked below each Primary function. The equivalent Tertiary and Correspondent levels in the Functional Basis are collapsed into a single Correspondent level below each Secondary function. The Correspondent groups consist of approximately 15-25 functions each. Table 6 is a complete list of the Primary and Secondary functions for the terms extracted from the patent database.

Table 6: Top tiers of expanded Functional Basis derived from the USPTO patent database.

Primary	Secondary
Branch	Divide
	Extract
	Clean
	Distribute
	Penetrate
	Remove
	Dilute
	Split
	Disperse
	Break
	Machine
Channel	Import
	Export
	Transport
	Transmit
	Translate
	Move
	Rotate
	Transfer
	Oscillate
	Arrange
	Direct

Primary	Secondary
Convert	Convert
	Substitute
	Transform
	Produce
	Condition
	Treat
	Modify
Connect	Connect
	Couple
	Mix
	Mount
	Apply
	Add
	Combine
	Encounter
	Bond
Control	Increase
	Decrease
	Decrement
	Shape
	Control
	Form
	Change
	Adjust
	Actuate

Primary	Secondary
Provision	Store
	Supply
	Prevent
	Collect
	Protect
	Stop
	Contain
	Inhibit
Signal	Indicate
	Process
	Display
	Detect
	Measure
	Compare
	Select
	Define
	Determine
	Monitor
	Output
	Calculate
Support	Place
	Secure
	Support
	Align
	Anchor
	Hold
	Maintain

The structure of the expanded Functional Basis vocabulary is 1700 unique functions organized into 74 groups of Secondary functions. The Secondary functions and associated correspondents are mapped into the eight (8) Primary functions. Table 7 illustrates the hierarchical structure for two of the Secondary functions: *divide* and *import*.

Table 7: Examples from the expanded Functional Basis vocabulary for the Secondary functions of *Divide* and *Import*

Primary	Secondary	Correspondents
Branch	Divide	Section
		Divide
		Segment
		Branch
		Sort
		Partition
		Tab
		Miss
		Diverge
		Fractionate
		Cube
		Segregate
		Dissociate
		Graduate
		Quantize
		Parse
		Allot
		Buck
		Dissect
		Dismantle
		Shred
		Interdigitate
		Packetize
		Compartmentalize
		Part
		Separate
		Digitalize
		Modularize
		Butcher
		Sectionalize

Primary	Secondary	Correspondents
Channel	Import	Permit
		Insert
		Input
		Introduce
		Inlet
		Accept
		Admit
		Fetch
		Inflow
		Breathe
		Aspirate
		Import
		Invite
		Ingest
		Invade
		Inhale
		Include
		Obtain
		Receive
		Enter
		Cannulate
		Induct
		Internalize
		Imbibe

A brief review of the Correspondent functions in Table 7 illustrates the potential for innovative solutions. For example, the process-specific functions of *shred* and *parse* primarily occur in disparate domains, material processing versus information processing. This research proposes useful design information can be mapped across these domains leading to novel solutions (Chan, et al. 2011). The same mapping of far-field analogies can be derived from a common generic function implemented across multiple domains such as *input* which has solutions in mechanical, electrical, and software high level domains.

Utilizing the structure of the function vocabulary, the patent search database is constructed by indexing additional patents against the completed function vocabulary. The whole of the patent database is too large to index given the current hardware and software prototype implementation. For the purposes of this research, a representative sample database is constructed from a subset of the USPTO patent database. Three continuous selections of 100,000 patents each are chosen to be indexed. The patent groups, plotted in Figure 23, are selected chronologically with the first selection from patents 3,560,000 to 3,660,000, the second selection from patents 5,000,000 to 5,100,000 and the final selection from patents 7,500,000 to 7,600,000 spanning the years from 1971 to 2009. After omitting repealed or missing patents, the sample database consists of approximately 275,000 patents mapped into document vectors resulting in an approximately 275,000 x 1700 patent vector matrix.

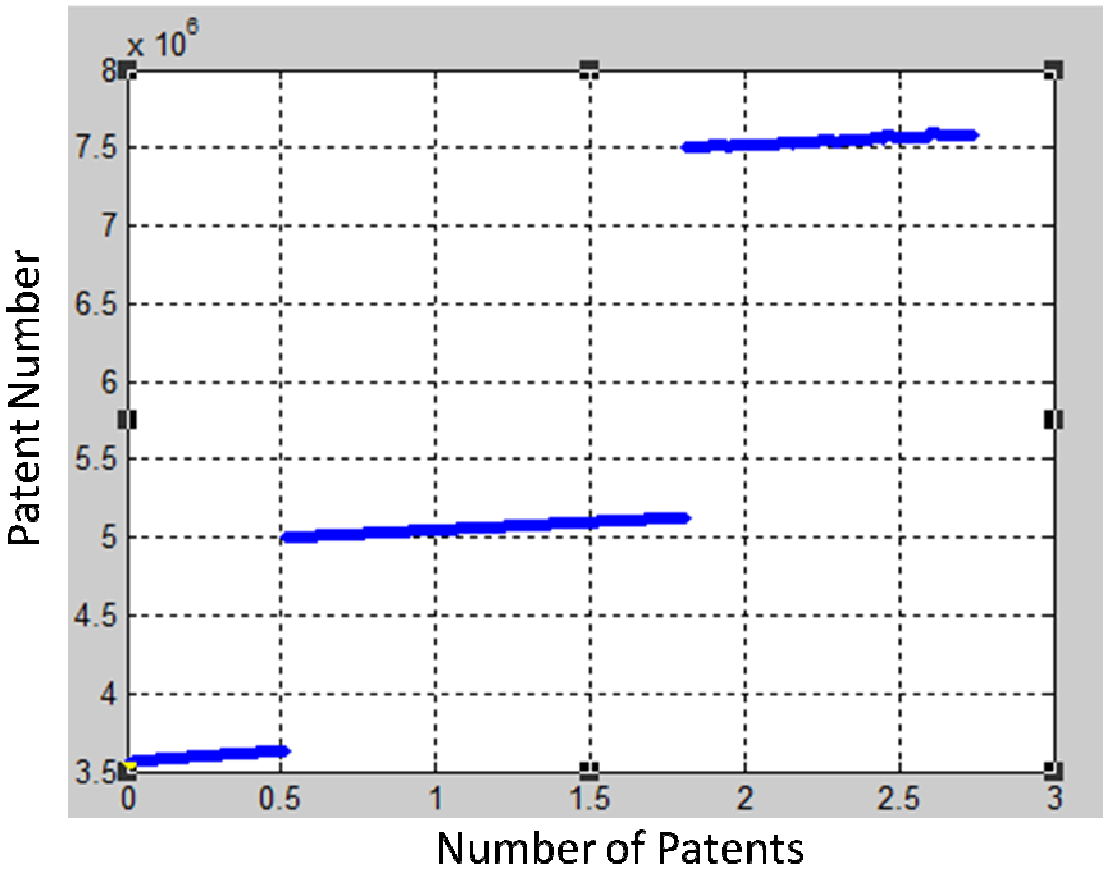


Figure 23: Patents selected to construct the patent vector matrix database spanning 1971 to 2009.

The document-vector mapping process uses several of the same algorithms as the initial indexing process. The workflow for mapping the patent to a document vector is given in Figure 24. First, the important information is parsed directly from the HTML source code including the patent title, primary class, abstract, claims and description. Next, part-of-speech tagging is applied to the full-text fields of the patent. The suffix and prefix stemming are also applied during this step. Finally, the function verbs are extracted per

the relevant POS tags discussed previously, and the terms are mapped to a binary document vector where *1* is the occurrence of the term and *0* is the absence of the term.

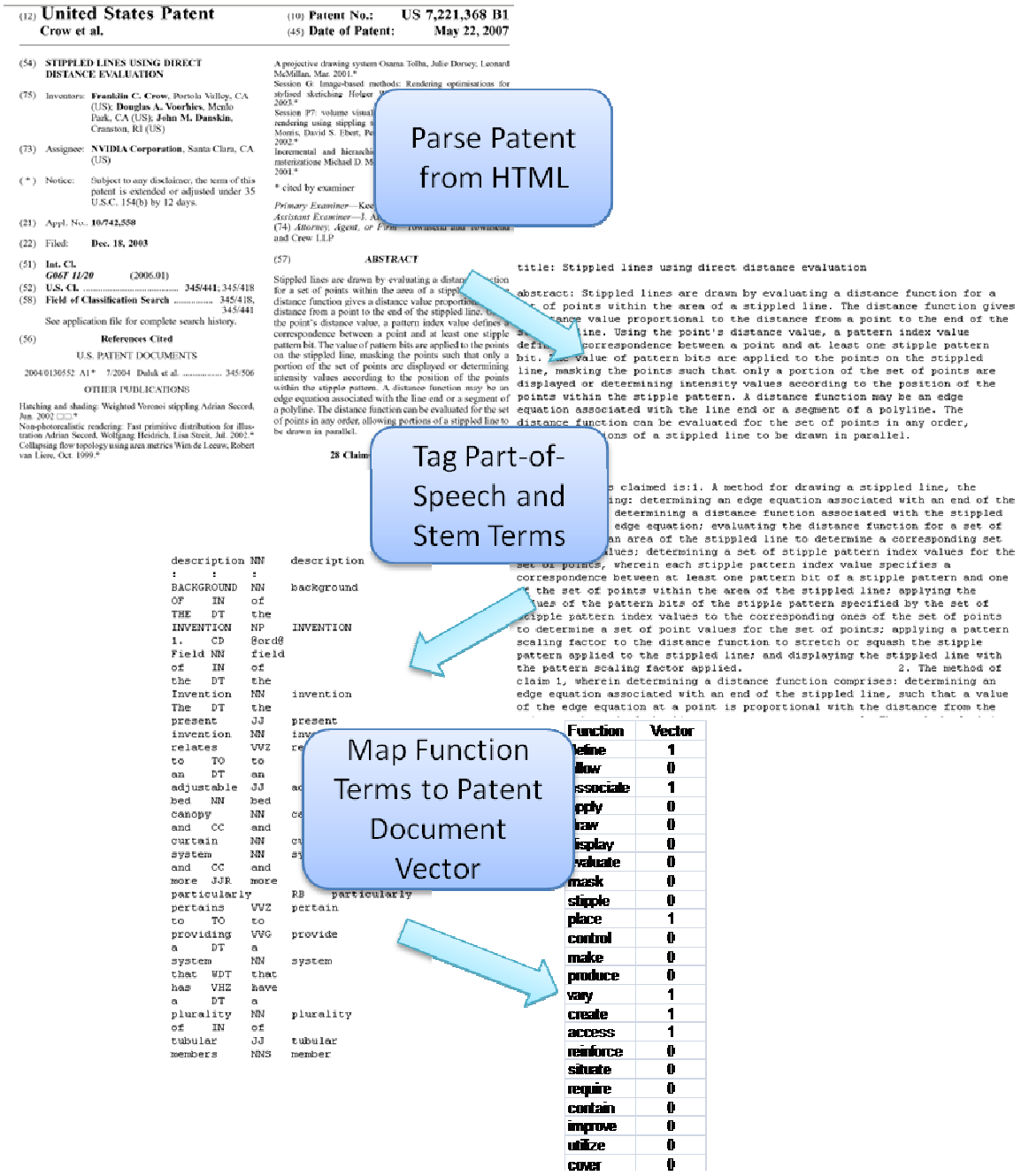


Figure 24: Workflow for generating patent document vectors from patent HTML text.

One significant trend that is identified by sampling patents chronologically is that the overall length of patents has increased significantly over time. The patents from the first group issued in the early 1970's are typically less than ten pages in length. Patents from the final group issued in 2009 are typically over twenty pages in length with many patents exceeding one hundred pages. The variation in length can have an adverse effect on the similarity scoring metrics which are discussed in the following section.

Query Generator and Search Result Viewer Interfaces

The binary document vector matrix contains both the functional content information for each patent as well as the term-document frequencies across all patents indexed. The term document frequency and the patent functional content are used to derive the similarity metric for ranking the search results. As discussed previously, the *document frequency* (*df*) is a common term weighting scheme and in particular the *inverse document frequency* (*idf*) is used to weight rare terms higher than common terms (Manning, et al., 2009, Salton and Waldstein, 1978). The *inverse document frequency* is given as

$$idf_t = log \frac{N}{df_t}$$

where N is the total number of documents and df_t is the document frequency of term t. Previous research has shown more specific function verbs can yield more novel solutions (Linsey, et. al, 2011), and the *idf* weighting yields a higher cosine similarity score for patents that contain process-specific functions. The *idf* is calculated for each term, and each element of the document vector matrix is scaled according to the calculated weight for that term. Furthermore, each document vector is normalized to generate a patent document unit vector matrix. The normalization is completed to simplify the cosine similarity calculation. The *patent functional content* (*fcm*) metric is a normalized measure

of the total functional content with a specific patent. The equation for the *fcm* metric is given as

$$fcm_k = \frac{total\ number\ of\ terms\ in\ patent_k}{total\ number\ of\ terms\ in\ database}$$

The *fcm* metric increases the weighting of patents with high functional content. The reasoning for including this metric is a hypothesis that functionally rich patents contain more information which can be mapped as analogies. The *total relevancy score* is then defined as a linear combination of the *idf*-weighted cosine similarity metric and the *patent functional content* metric as summarized in Table 8.

Table 8: Metrics for calculating similarity between the patent-document and query vectors.

• Query-Patent Cosine Similarity	$\cos\theta = \frac{Query \bullet Patent}{\|Query\| * \|Patent\|}$
• Patent Functional Content Metric	$FCM = \frac{\sum Patent_{term(i)}}{NumTerms}$
• Total Relevancy Criteria	$Score = \alpha \cdot \cos\theta + \beta \cdot FCM$

The alpha, α, and beta, β, coefficients are tuning parameters used to bias the relevancy ranking towards a higher weighting on either the cosine similarity or functional content metrics. The tuning parameter weights are determined through a parametric evaluation process by running multiple patent searches using the Query Generator and Search Results Viewer interfaces described in the following section.

The Query Generator tool is created to automate the process of constructing the patent query vector. The GUI builds the query using the expanded Functional Basis vocabulary hierarchical structure as shown in Figure 25. First, the user selects the Primary function corresponding to the high-level functionality derived from the

functional model of the design problem. Next, the user selects a Secondary function corresponding to the specific functionality that will be retrieved. Selecting the *More* button will display all of the Correspondent functions organized under the Secondary function. Once the Secondary function is selected, the interface populates the query vector with all Correspondent terms associated with the Secondary function. The one-to-many term mapping eliminates the issues related to synonymy encountered in the Boolean search methods discussed previously.

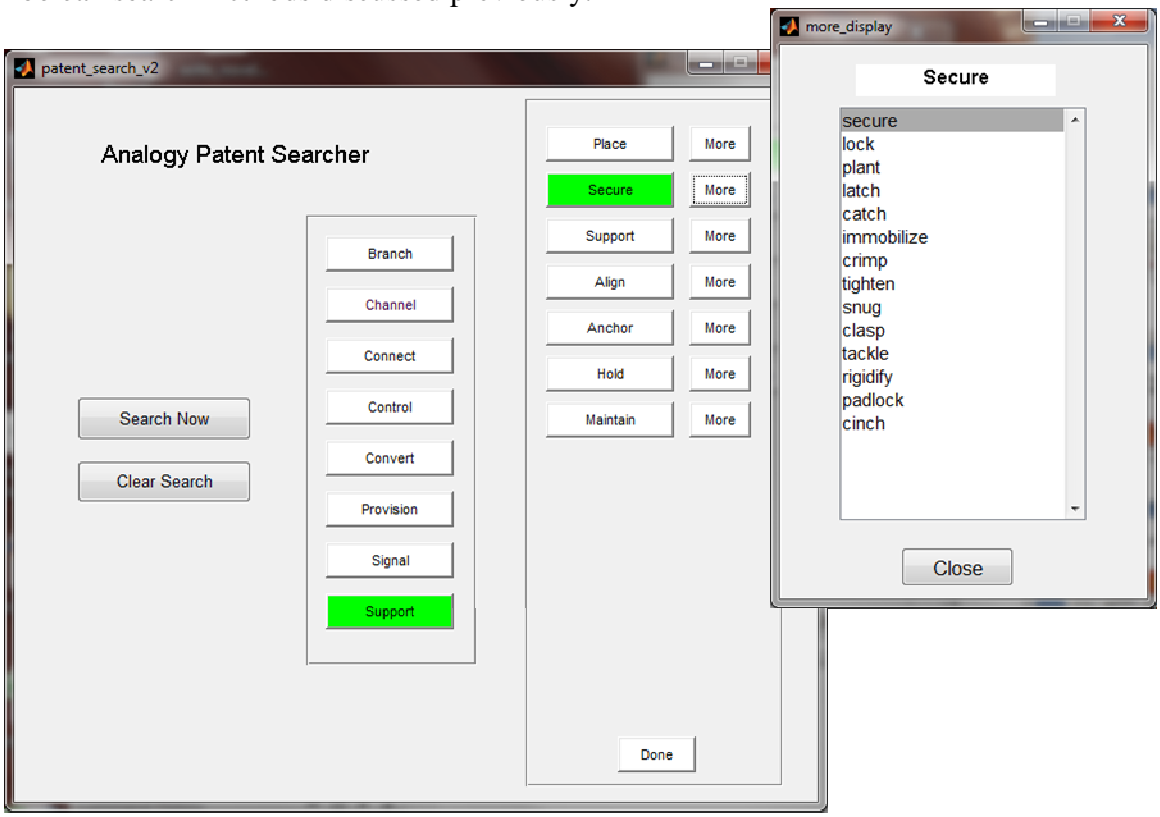


Figure 25: Query Generator user interface

Additional Secondary functions can then be selected to further populate the query vector for a particular Primary function. The *Done* button is selected to save the new query vector once all Secondary functions are chosen. The process can then be repeated for

additional Primary functions. The complete code for the Query Generator is located in Appendix B.

Once the query construction is complete, the Search Now button will launch the Search Result Viewer. The Viewer in Figure 26 performs multiple functions including calculating the *cosine similarity*, *fcm*, and *total relevancy score*, extracting the top results and clustering the results by patent class.

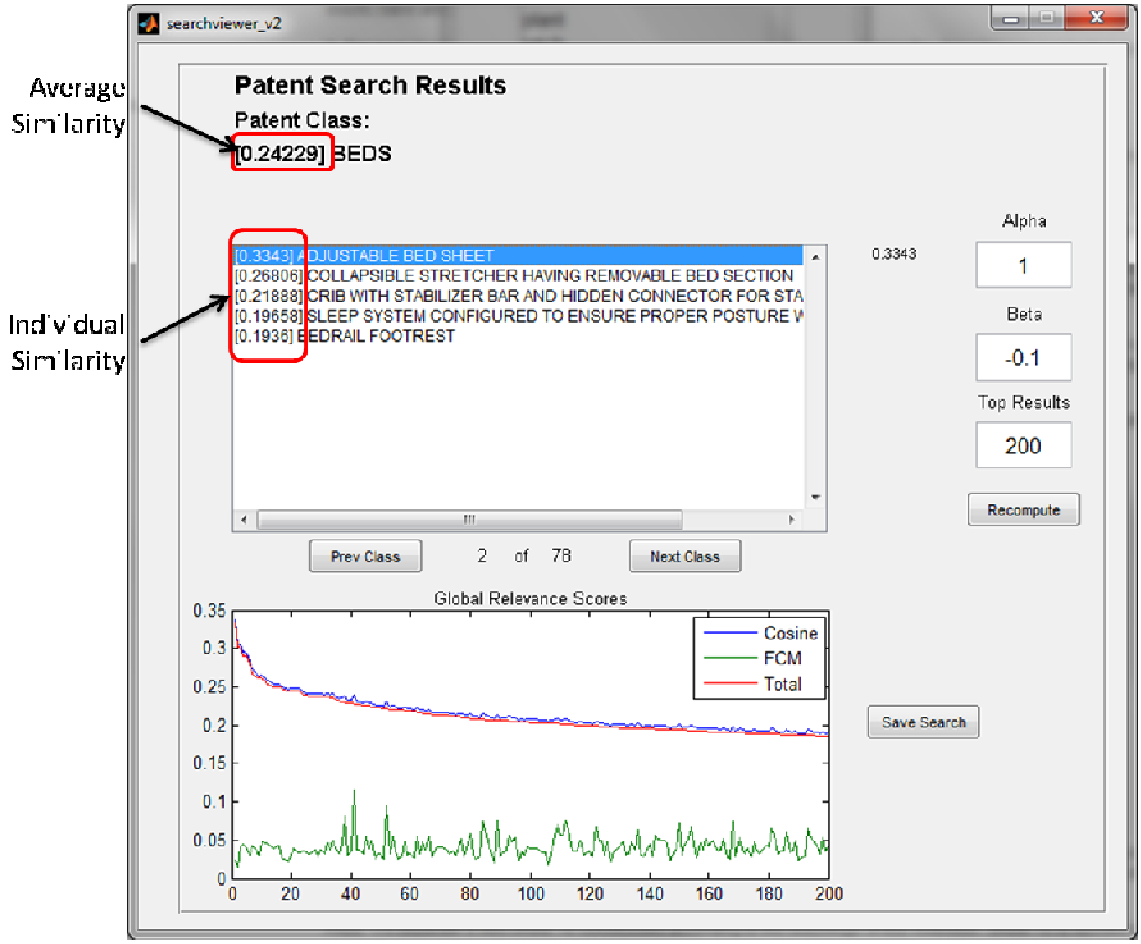


Figure 26: Example search result for the query *Support→ Secure.* First column of search results indicates the similarity score for the specified α and β coefficients.

The cosine similarity is calculated for all documents simultaneously by first normalizing the query vector to form the query unit vector and then calculating the dot product of the unit query vector with the document vector unit matrix using the equation

$$\overrightarrow{cos}_{similarty} = q^T \cdot d$$

where $\overrightarrow{cos}_{similarty}$ is a vector containing all cosine similarity scores for the dot product of the query vector, q, and the document vector matrix, d. The total relevancy vector is calculated by the linear sum of the cos_{vector} and the *functional content metric* vector weighted by the user-defined α and β coefficients, respectively. The top n results as specified by the user are retrieved, sorted by total relevancy score and clustered by primary patent classification. The *Previous* and *Next Class* button enable the user to quickly scroll through the search results. The similarity scores for the individual patents are clearly indicated in the first column of the results list. The average relevancy score for the patent class is given before the title to help the user quickly identify patent classes with high potential for identifying functionally relevant patents. The complete code for the Search Result Viewer is located in Appendix B.

Selecting one of the search results automatically opens a web browser window with a PDF version of the selected patent. The search viewer uses Freepatentsonline.com as the web interface. The PDF version is displayed due to the fact the patent illustrations are included as opposed to the text-only version. An example of a search result for the query *Support→ Secure* from Figure 26 is illustrated in Figure 27. The high relevancy score is due to the patent containing the terms *secure, lock, catch, tighten, snug,* and *cinch* which are six of the fourteen functions in the search query. The *adjustable bed sheet* patent solution for *Support→ Secure* could be readily mapped as a solution to domains beyond bedding. In Chapter 4, the patent search method is applied to two case

studies using real-world design problems to illustrate how the analogy mapping is applied.

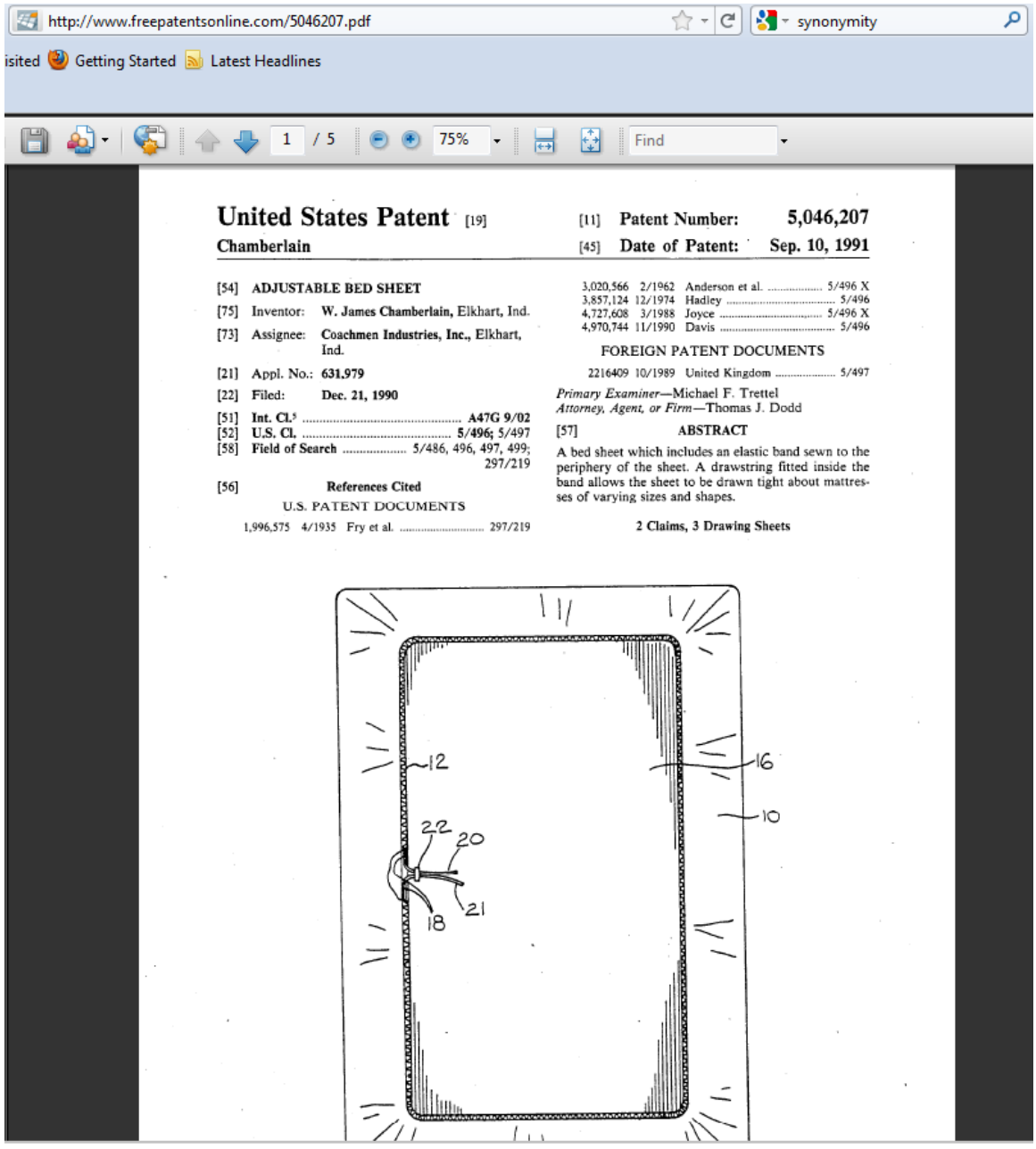


Figure 27: Adjustable bed sheet patent (5,046,207) search result for the query *Support→ Secure* containing matches for query terms *secure, lock, catch, tighten, snug,* and *cinch.*

To determine the optimal weighting for the total relevancy score coefficients, several searches are conducted over various function combinations. The Search Viewer interface enables the coefficients to be varied in real-time for the same search query allowing for multiple iterations for the same function query. Following a trial-and-error process where β is varied from 1 to -1 keeping $\alpha = 1$, the search results provided more functionally relevant results for negative values of the *fcm* coefficient. This result contradicts the original hypothesis proposed which stated functionally rich patents are more readily mappable to functional analogies. In practice, positive values of β skew the results towards long patents since, statistically, patents which contain more text will contain more function verbs. Elucidating useful analogies from these broad patents is cognitively more difficult than functionally focused patents. The default values for α and β are set to 1 and -0.2, respectively, which focused the total relevancy score toward functionally focused patents. Future work regarding optimization of the search tuning parameters will included an experimental study to independently verify this finding.

CONCLUSIONS

The Vector Space Model-based patent search engine provides an organized method for identifying functionally similar patents independent of the patent solution domain. The domain-independent search capability is achieved through the systematic derivation of a complete functional vocabulary extracted from the target knowledge base of the USPTO patent database. Several natural language processing algorithms are developed and implemented to indentify a finite set of function verbs and the functions are organized into an expanded Functional Basis vocabulary with a hierarchical structure. The 1700 function terms are utilized to generate a searchable document vector matrix consisting of approximately 275,000 patents. Search tools such as the Query Generator

interface and the Search Result Viewer interface are created to enable effortless access to the vast design information contained in the limited sample of the patent database. Additional insight gained in model development is patents that are more functionally rich are more difficult to map analogically due to the broad functional coverage. Further research is needed to verify the optimal values for the total relevancy score metric beyond the trial-and-error process undertaken by the researcher. Other proposed improvements to the search engine implementation include expanding the patent coverage with the ultimate goal being indexing the entire patent database.

Chapter 4: Patent Analogy Search Methodology and Case Studies

Two design problems are chosen to test the efficacy of the VSM patent search tool for augmenting concept generation. The first case study seeks to reproduce and improve upon the classic Design-by-Analogy problem of the guitar pickup winder (McAdams and Wood, 2002). The second case study is an original design problem describing an automated device for cleaning windows. In Chapter 3, the expanded Functional Basis vocabulary, patent document-vector matrix, query generation and search tools are developed. In this chapter, those tools are combined into a structured methodology for identifying analogous patents. The product design workflow in Figure 28 illustrates the interaction of patent analogy search within the product design methodology.

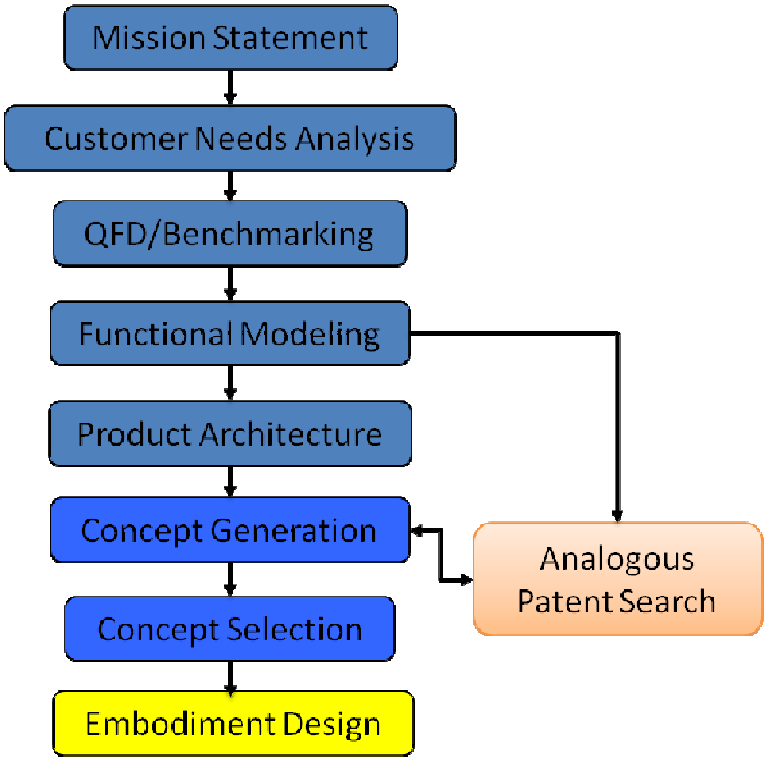


Figure 28: Product design workflow with patent search tools augmenting the concept generation process.

The analogy search tool is used as a supplemental technique to traditional concept generation methods such as brainstorming, brainsketching, and the CSketch/6-3-5 method (Osborn, 1957, Vangundy, 1988, Otto and Wood, 2001, Markman and Wood, 2009). The device functionality described during the functional modeling phase of the design process is used directly to create functional semantic representations of the design problem which are independent of flow domain as illustrated in Figure 29.

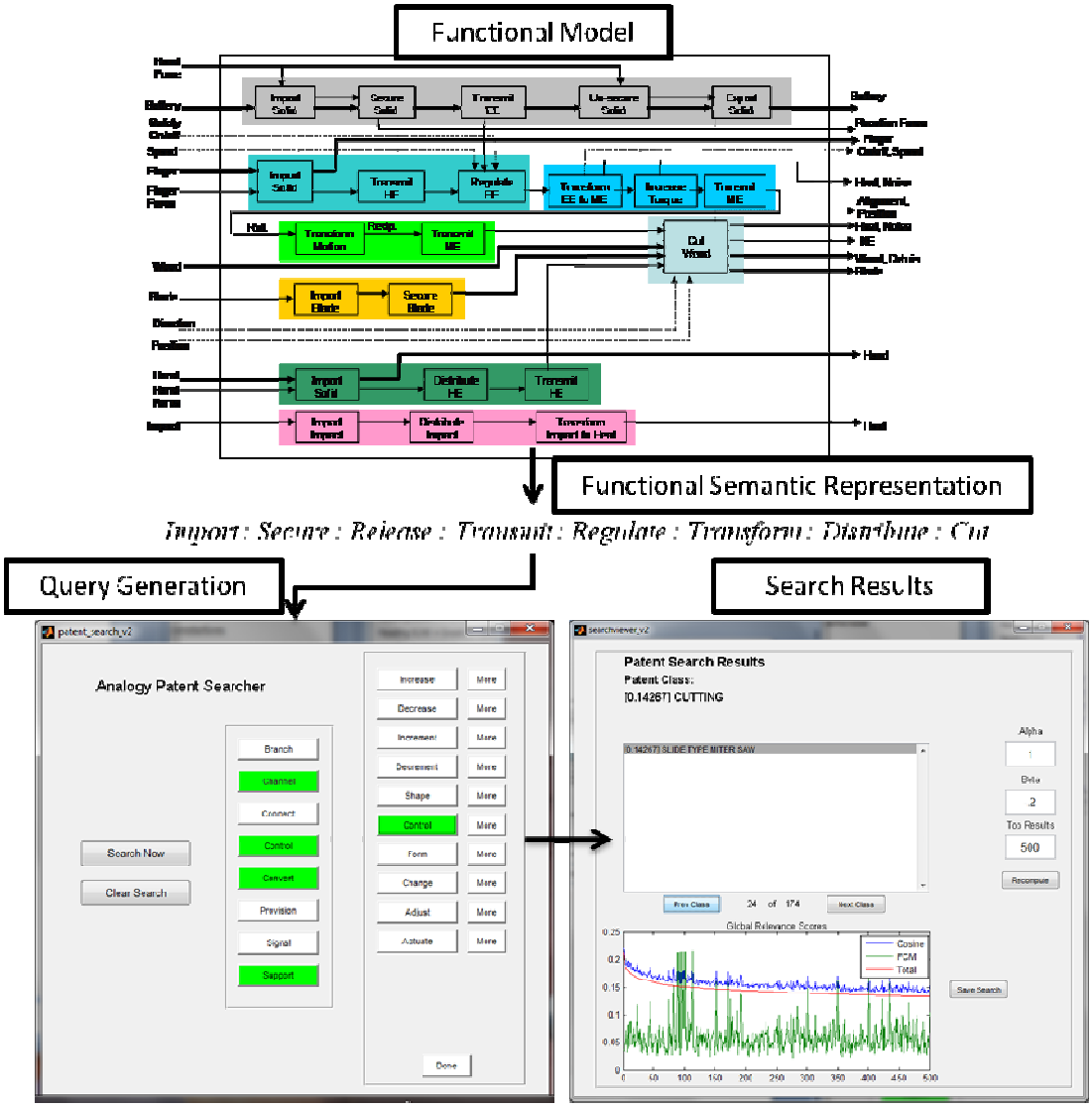


Figure 29: Analogy search methodology from functional model to patent search results.

The functional semantic representations are mapped to the expanded Functional Basis vocabulary, and the Query Generation tool is utilized to create the query function vector for the device. The Search Result Viewer identifies the functionally similar patents where analogies to the design problem likely exist. The final step in the process is mapping useful patents back into the original problem domain. In the following sections, the analogy search methodology outlined above is applied to the two case studies.

GUITAR PICKUP WINDER

The first case study was chosen to illustrate that the Patent Analogy Search Engine could be utilized to reproduce, if not improve on, the solutions for the classic Design-by-Analogy problem of the guitar pickup winder. McAdams and Wood (2002) compare the functionality of the pickup winder, which is used to manufacture electromagnetic coils for electric guitars, to a database of 68 products. The five most functionally similar products are shown in Table 9 where λ is the normalized similarity index.

Table 9: Results of similarity calculation for the pickup winder (McAdams and Wood, 2002)

Product	Similarity Index, λ
Pickup winder	1.0
Fruit & Vegetable Peeler	0.78
Electric can opener	0.74
Electric sander	0.73
Fishing reel	0.72
Cheese grater	0.69

In Figure 30, a simplified functional model of the pickup winder is given which includes the top six functions as determined from the weightings of the corresponding customer needs.

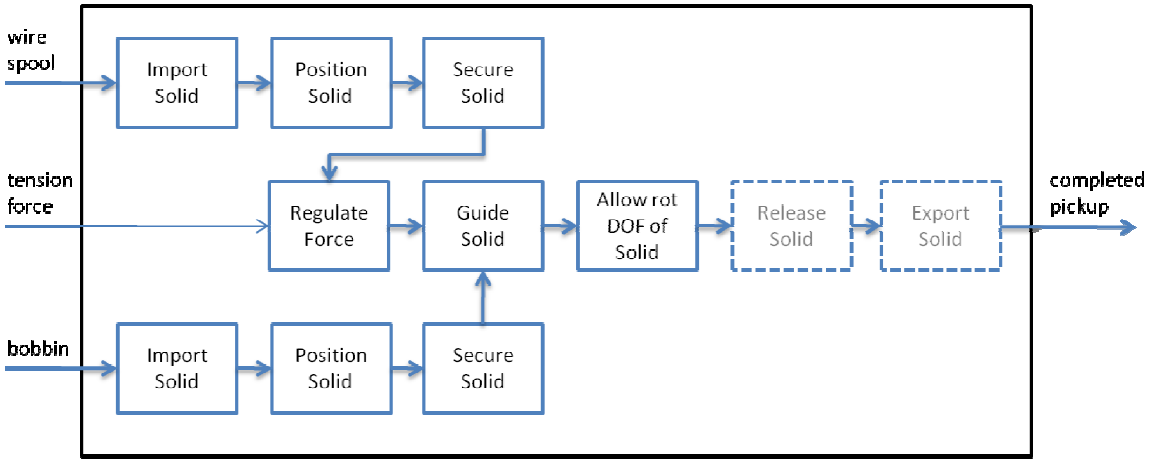


Figure 30: Simplified functional model of a guitar pickup winder

The generic top six functions are: *import*, *secure, position*, *regulate*, *guide*, and *allow rotational DOF* as shown in the solid boxes. In the expanded patent Functional Basis, the functional semantic representation maps as follows:

- Import: Channel → Import
- Secure: Support → Secure
- Position: Support → Place
- Regulate: Control → Control
- Guide: Channel → Direct
- Allow rotational DOF: Channel → Rotate

After the mapping from the original functional model to the expanded basis is established, the search for analogous patents is performed utilizing relevance ranking weights of *alpha* = 1 and *beta* = -.2 and the top 10000 results are reviewed.

The first phase of the case study was to determine whether the patent search could extract the analogous products from Table 9. Figure 31 shows the example of the search result for the fruit peeler. Considering the relative sparsity of patents included in the prototype database (~6% of electronically available patents), the search results are very successful with the search coverage including three of the five top analogous devices.

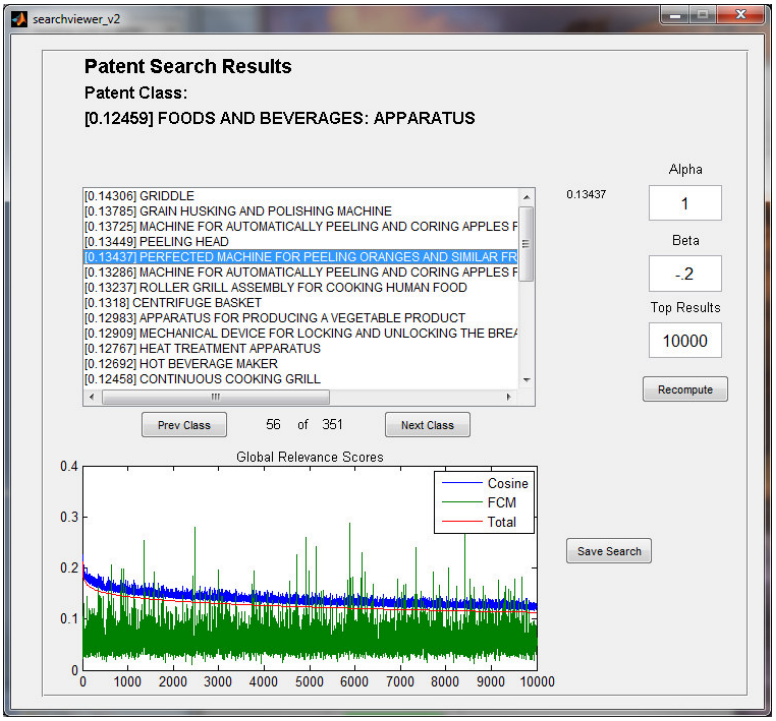


Figure 31: Search results for pickup winder generic functions showing fruit peeler analogy

Figure 32 depicts sample illustrations for the three analogous devices retrieved: a fruit peeler, a fishing reel with disengageable spool, and a belt sander.

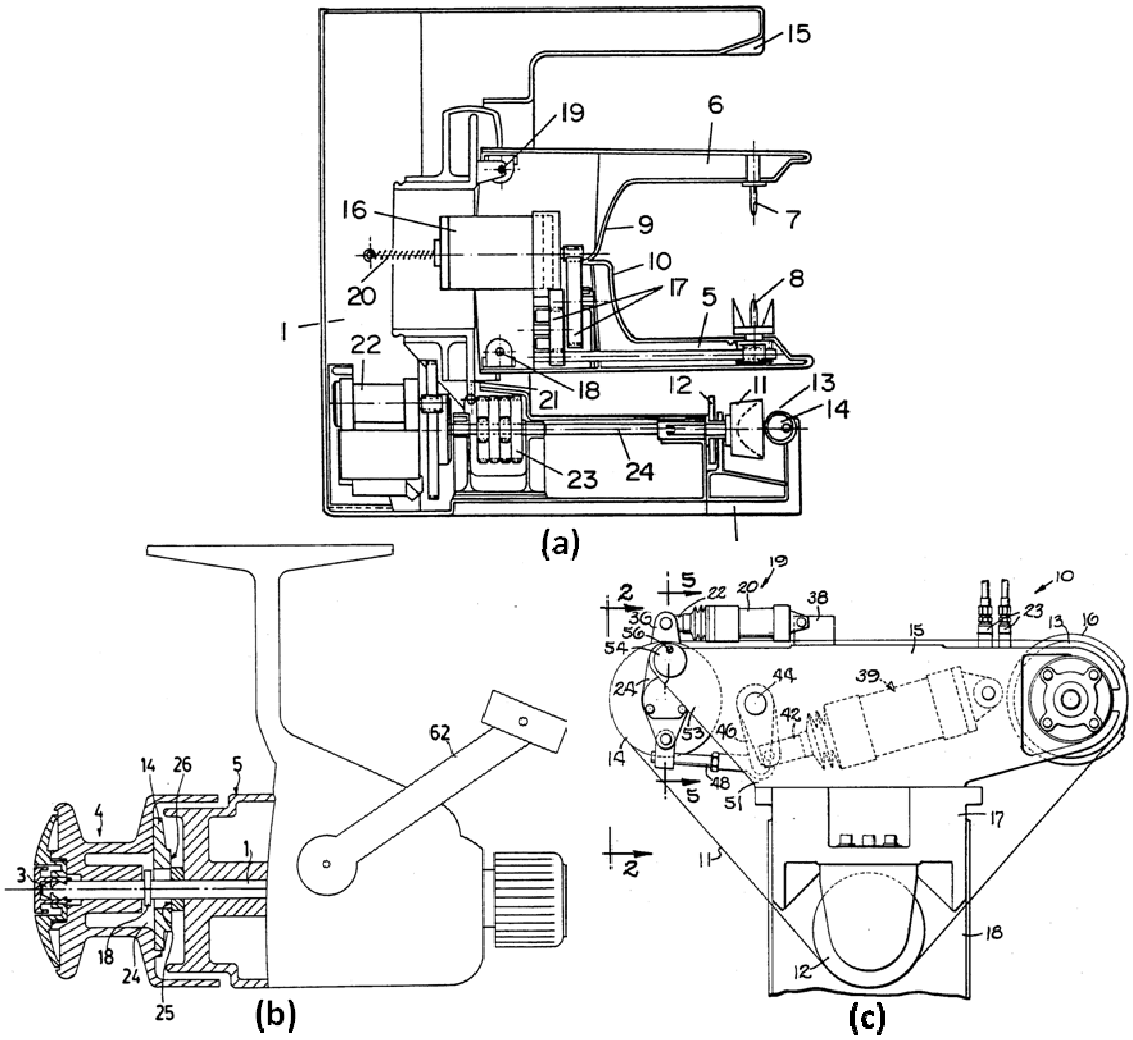


Figure 32: Illustrations from analogous patents for the pickup winder: (a) US patent 5105735: Perfected machine for peeling oranges and similar fruits, (b) US patent 5121888: Spinning reel with a spool disengageable from a rear-mounted drag, (c) US patent 3577684: Abrading machine with steering roll and tensioned abrading belt

It must be noted that a large pool of search results is required to identify the three analogous patents. The spinning reel and belt sander occurred within the top 1000 search results, but the fruit peeler is not retrieved until a group of the top 10000 results are extracted. The large number of patents required to extract the fruit peeler analogy is

caused by a highly populated query vector resulting from multiple Secondary functions used in the query generation. With each Secondary function mapping to an average of 20 Correspondents, the total number of terms in the query vector is approximately 120 terms and leads to poor resolution with respect to the cosine similarity metric. One of the significant insights gained through this case study is multiple searches on individual functions improves discrimination among the search results. Despite the relevancy resolution issues encountered, an additional patent was found that, if implemented into the pickup winder design, would provide a novel means of controlling the wire tension.

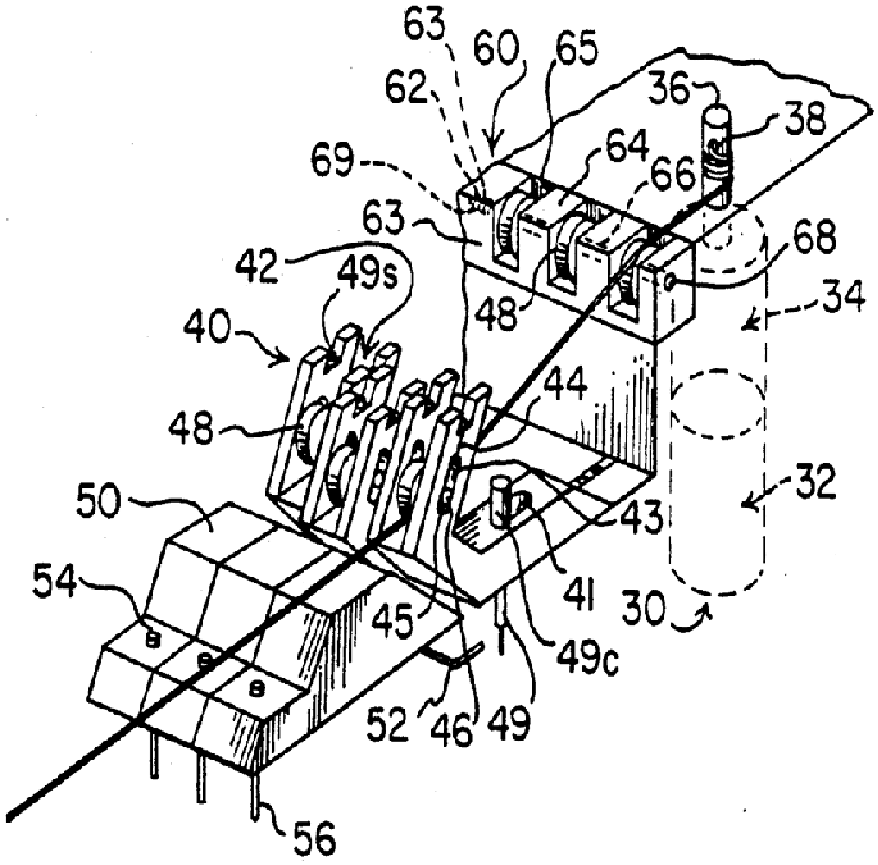


Figure 33: Wire tension control analogy solution, Patent 5,038,657

In Figure 33, a tension control mechanism for a musical instrument excites the wire with a known frequency and measures the response. The tuning device automatically adjusts the tension to bring the frequency response to within the specified range. A device using piezoelectric actuators to control the tension, actuate the wire and sense the frequency response can be designed into an advanced version of the pickup winder if constant tension is critical for enhanced pickup performance or process control consistency.

WINBA- THE AUTOMATED WINDOW WASHER

The second case study utilized to evaluate the Patent Analogy Search Engine is an automated window washing device. The design problem is to design a self-contained window cleaning device. Once initialized, the device will begin an automated routine for removing dirt, film, and debris from the window surface without user interaction. The general problem statement allows for multiple process choices such as the power source and cleaning method. The blackbox functional model for a battery-powered device which utilizes a liquid media for cleaning is shown in Figure 34.

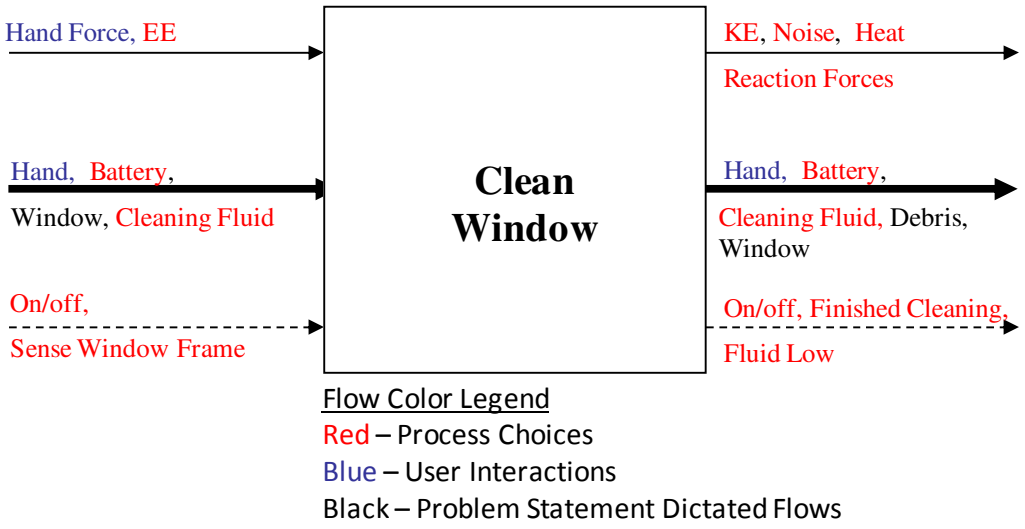


Figure 34: Blackbox model of automated window washer

Other alternative process choices for a power source are solar-power and fuel cells among others. Alternative cleaning method process choices omit the cleaning fluid and rely on mechanical removal of debris. The functional model for the battery-powered, fluid-based window washer is given in Figure 35.

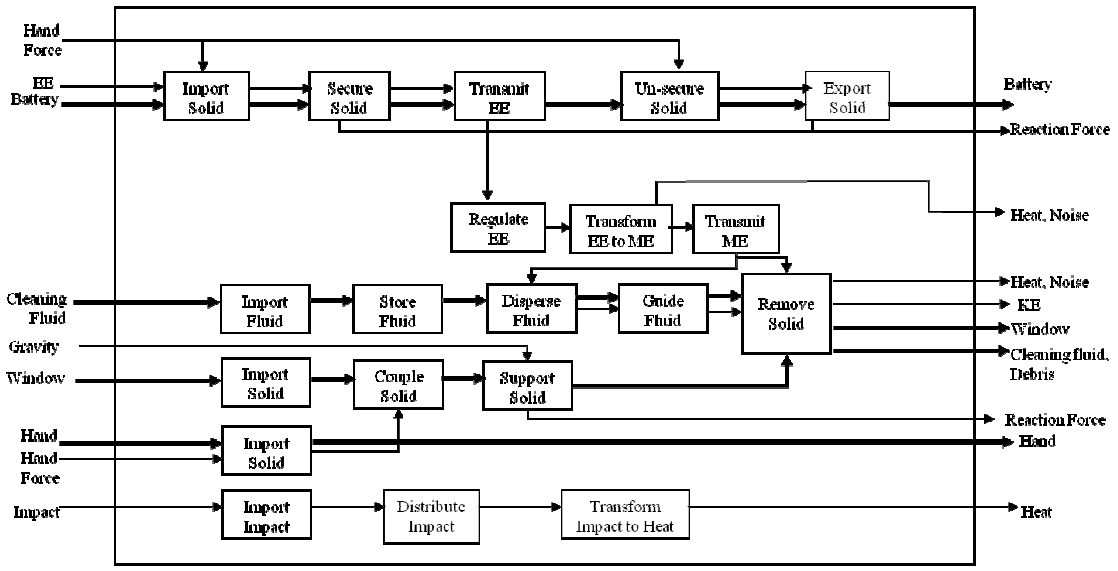


Figure 35: Functional model for a battery-powered window washer using a fluid cleaning media

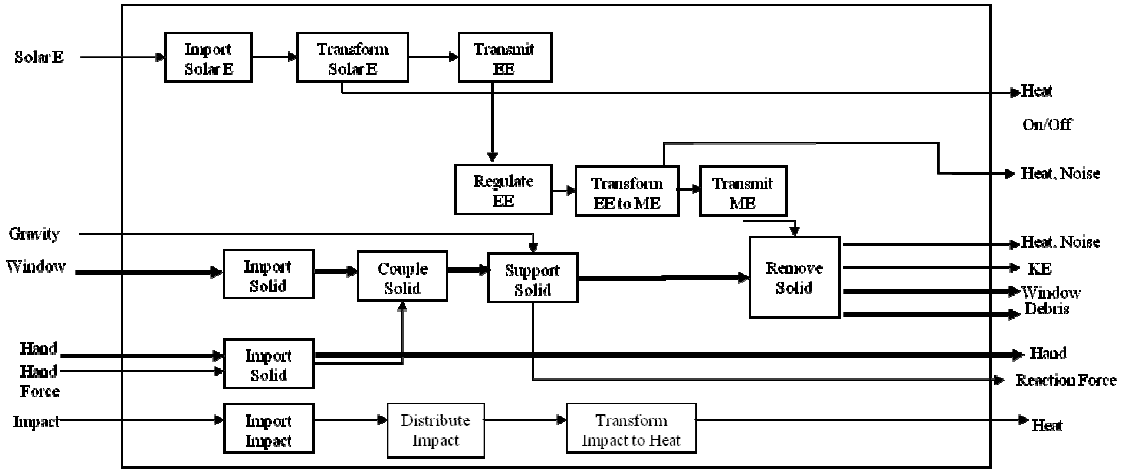


Figure 36: Functional model for a solar-powered window washer using mechanical cleaning methods

An alternate functional model for a solar-powered window cleaning device which uses a mechanical debris removal system is shown in Figure 36. Both devices share the same core functionality which is coupling the device to the window, converting some type of energy to mechanical motion, and removing the debris from the surface of the window. From the common core functions, a simplified functional model of the device is generated as shown in Figure 37.

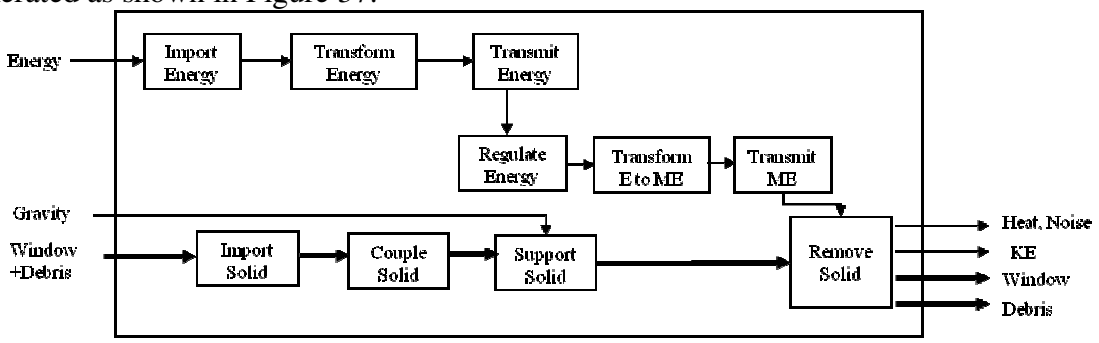


Figure 37: Simplified functional model of core functionality for an automated window washer.

The functional semantic representation of the simplified model becomes

Import : Transform : Transmit : Regulate : Couple : Support : Remove

Further generalizing the model into the Primary functions results in the functional semantic representation given as

Channel : Branch : Convert : Control : Connect : Support

A separate analogy search is performed for each Primary function using the Secondary functions most relevant to the original design problem. The multiple search approach is used to maximize the relevancy score resolution for each query to mitigate the large query vector resolution issue identified from the pickup winder case study. The Secondary functions utilized for each search query are:

- Channel → Import, Transmit and Translate

- Branch → Remove, Clean and Disperse
- Convert → Transform and Treat
- Control → Control and Adjust
- Connect → Connect, Mount, and Couple
- Support → Secure and Align

All searches are performed using the default values for the total relevancy score metric of $\alpha = 1$ and $\beta = -.2$. The top 500 results are retrieved for each search. Table 10 summarizes the relevant patents compiled from the search results for the queries listed above.

Table 10: Combined search results for the automated window washer

Patent Title	Patent Number
[0.19507] METHODS FOR CLEANING MATERIALS	7556654
[0.15162] AUTONOMOUS FLOOR-CLEANING ROBOT	6883201
[0.14184] SWIMMING POOL VACUUM CLEANER WITH ROTARY BRUSH	5044034
[0.13631] DEVICE FOR CLEANING A WINDOW GLASS	5086533
[0.1916] POWERED CLEANER/POLISHER	7565712
[0.13631] METHOD AND APPARATUS FOR CRYOGENIC REMOVAL OF SOLID MATERIALS	5025632
[0.15685] WASHING DEVICE FOR CLEANING A CYLINDER OF A PRINTING MACHINE	5035178
[0.13624] METHOD OF POLISHING A SEMICONDUCTOR WAFER	7559825
[0.18867] VEHICLE WASHING APPARATUS	5077859
[0.15076] APPARATUS FOR SUPPORTING A DIRECT DRIVE DRILLING UNIT	5038871
[0.15398] CONSTRUCTION ELEVATOR ASSEMBLY	5033586
[0.17476] METHOD AND SYSTEM FOR MAINTAINING EQUAL AND CONTINUOUS FLOWS OF LIQUID TO AND FROM INTERMITTENTLY OPERATING APPARATUS	3589389
[0.16747] ELECTRICALLY CONTROLLABLE WINDOW TREATMENT SYSTEM TO CONTROL SUN GLARE IN A SPACE	7588067
[0.1792] PROCESS FOR THE CONTROLLED RELEASE OF METERED QUANTITIES OF LUBRICANT WHEN COATING PRESSING TOOLS WITH LUBRICATING LIQUIDS AND SUSPENSIONS AND APPARATUS FOR CARRYING OUT THE PROCESS	3589389

The patent found for the window cleaning device (Patent 5,086,533) is a very near-field analogy to the proposed design problem. The device shown in Figure 38 utilizes a squeegee mechanism with a fluid application system to automatically clean windows.

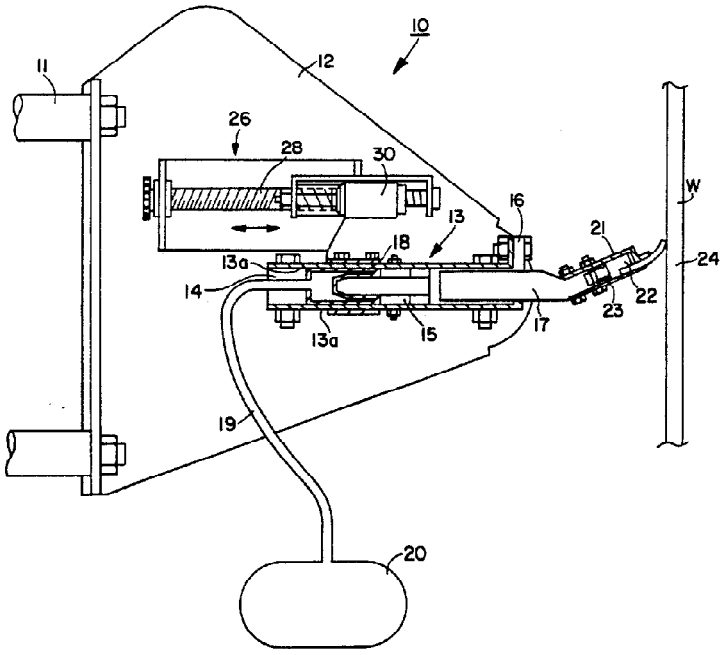


Figure 38: Automated window cleaning device found using the analogy search engine is an example of a near-field analogy (Patent 5,086,533).

A second cleaning device, shown in Figure 39, is used for automatically cleaning floors.

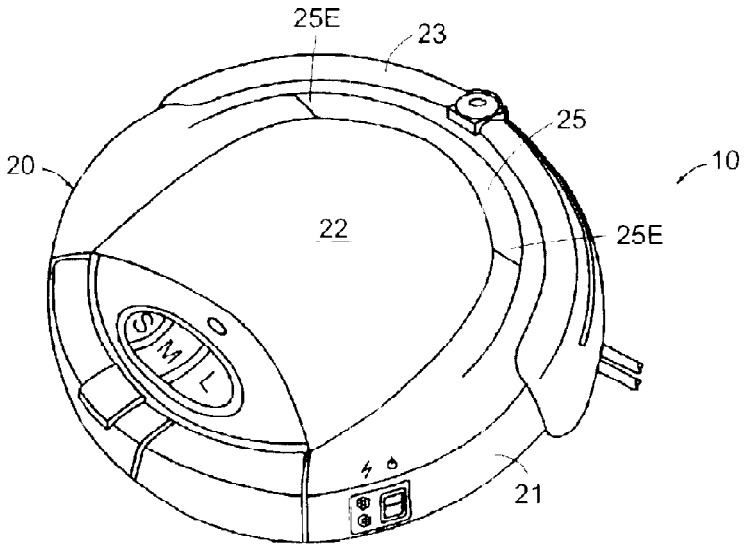


Figure 39: Automated floor cleaning device found using the analogy search engine is an example of a far-field analogy (Patent 6,883,201).

The floor cleaning robot (Patent 6,883,201) solution, better known as the iRobot Roomba™, performs the same desired functionality as the automated window washer, but the application is in a different domain (floors versus windows). Therefore, this solution is a far-field analogy that is readily adaptable to the window cleaning domain. The missing functionality of coupling the device to a window can be derived from other far-field analogies such as the wafer polishing patent (7,559,825) which utilizes vacuum to couple the device to the wafer surface. A purely mechanical means of traversing vertical surfaces is described in Patent 5,033,586 for a transportable construction elevator, shown in Figure 40, using a pulley mechanism.

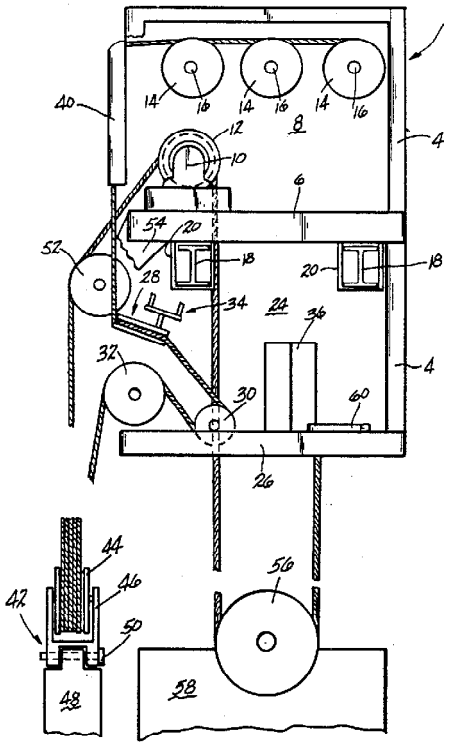


Figure 40: Transportable elevator system for vertically traversing buildings under construction (Patent 5,033,586).

Finally, entirely novel methods of cleaning surfaces are identified using the analogy search tool. Patent 5,025,632 describes an innovative process for cleaning surfaces utilizing a combination of cryogenically cooled fluids and mechanical abrasion. Although the cryogenic solution may not be feasible in applications of cleaning glass surfaces, the purpose of the tool is to stimulate novel problem solving by identifying both near and far-field analogies.

CONCLUSIONS

The Patent Analogy Search Engine and methodology performed well across both case studies. The first case study on the guitar pickup winder is used to verify the efficacy of the search procedure to reproduce a well known Design-by-Analogy solution. The search process is capable of extracting three of the five top analogous products for the pickup winder even given the limited patent database coverage. The pickup winder case study also illustrated the limitations of generating large query vectors due to the detrimental effect on the resolution of the relevancy metric.

The second case study applied the search methodology to the design problem of the automated window washer. Using lessons learned from the pickup winder, multiple search queries are generated to mitigate the similarity metric resolution problem. Six individual searches are performed and the compiled results include both near and far-field analogies. Among the far-field analogies are novel solutions for coupling the device to vertical surfaces using vacuum or transportable pulley systems and for removing debris using cryogenic fluids. In Chapter 5, an experimental study is conducted to evaluate the analogical search engine effectiveness on the quantity of ideas generated and the overall novelty of those ideas.

Chapter 5: Experimental Evaluation of the Patent Search Engine for Concept Generation Improvement

The case studies performed utilizing the analogy-based search engine show that both near and far field analogies can be derived from the patents obtained. The mapping of the analogies is relatively clear when the link is known *a priori*. The next step in the methodology development is to evaluate the effectiveness of the Patent Analogy Search Tool on a real-world design problem when the analogical mapping is not known beforehand. The experiment outlined in the following sections is designed to elucidate the effects of presenting functionally analogous patents during concept generation on the quantity and novelty of design solutions.

EXPERIMENTAL METHOD

An experiment is conducted to evaluate the efficacy of the Patent Analogy Search Tool to complement the concept generation phase of the design process. The first factor that will be investigated is the overall effect of augmenting brainstorming, or other ideation methods, by presenting functionally analogous patents derived from the search engine on the quantity and novelty of ideas. The second factor that is investigated is the effect within the analogy groups of searches derived from different levels of functionality, for example focusing on a single sub-function versus all sub-functions. Three levels of functionality are chosen for the analogy groups as shown in Table 11.

Table 11: Functionality level of analogy distribution among experimental groups

Group	Functionality Level of Analogies
Control	None
Analogy Group 1	Single Sub-Function
Analogy Group 2	Sub-Function Pair
Analogy Group 3	All Sub-Functions

The analogy and control groups all executed a three-phase ideation process. Phase 1 consisted of a 10 minute concept generation process which was common for all experimental groups. The differentiation between the analogy and control groups occurs during Phase 2 of the experiment. During this phase, the analogy groups are presented with the analogous patents according to the assigned functionality levels. During the same phase, the control group is given an article to review that is unrelated to the design problem to serve as a distracter. A primary purpose of this study is to test the cognitive effect of introducing analogous patents versus the unrelated distracter document on the concept generation process. Phase 2 was followed by a second 10 minute concept generation phase to record any additional unique solutions. The experimental workflow for all groups is illustrated in Figure 41. All groups are given the same total length of time for concept generation.

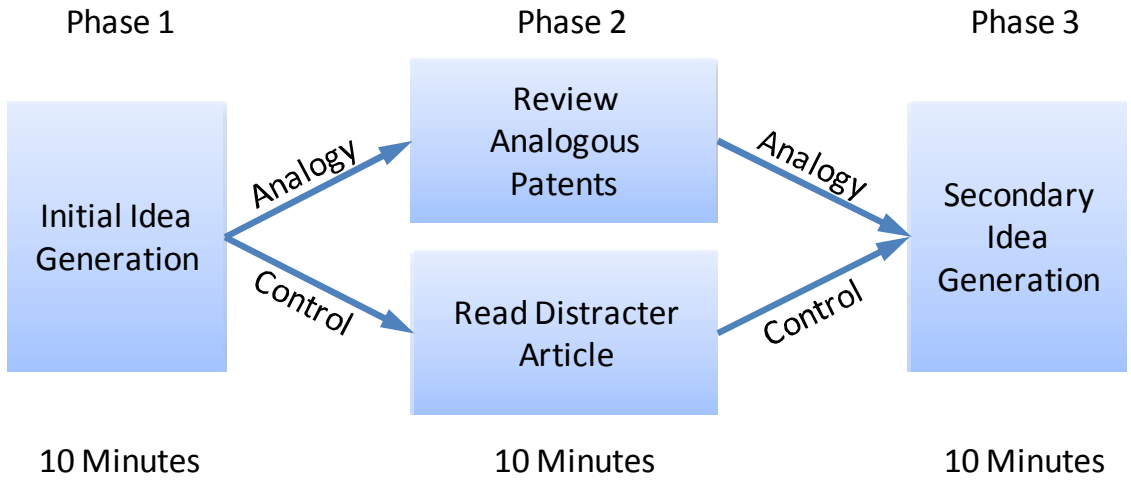


Figure 41: Experimental workflow comparison for analogy vs. control groups

Participants

The participants were senior mechanical engineering students enrolled in the design methods course at The University of Texas at Austin. Senior students were chosen

because they are of similar educational and experiential backgrounds. They have also had exposure to a wide variety of mechanical engineering theory and practical experience through design projects, internships, coops, etc. The relative uniformity across education and knowledge will minimize the variation between individuals during concept generation due to prior experience. As a result, main and secondary effects of the experimental conditions will be more readily identifiable. The total number of participants in the experiment was 68 students randomly assigned to one of the experimental groups discussed above resulting in 4 groups of 17 students each.

All participants were given the design problem described in the next section and worked individually to generate concepts. The students were instructed to record all concepts utilizing both words and sketches with distinct solutions recorded individually, similar to the Brainsketching process (Vangundy, 1988).

Design Problem Description

The design problem is to design a device to collect energy from human motion. The mechanical energy from human motion must then be converted to electrical energy and stored for later use to power small devices such as a radio or lighting device. Additional constraints on the design are:

- Low cost
- Easy to manufacture
- Portable

The complete problem statement is given in Figure 42. No further constrains or clarification regarding the design embodiment was given. This problem was chosen because it is a real-world, need driven problem with great breadth of possible solutions that a mechanical engineer with the participants' knowledge base would be able to solve.

Design Problem Statement
Design a device to collect energy from human motion for use in developing and impoverished rural communities in places like India and many African countries. Our goal is to build a low-cost, easy to manufacture device targeted at individuals and small households to provide energy to be stored in a battery. The energy is intended to be used by small, low power draw electrics, such as a radio or lighting device, hopefully leading to an increase in the quality of life of the communities by increasing productivity, connection to the outside world, etc.

Phase 1
Generate as many solution concepts to the design problem as you can. Record all concepts, including novel and experimental ones. You may use words and/or sketches to describe your ideas. Please record each distinct solution concept in the separate boxes provided. Additional pages are available upon request.

Figure 42: Design problem statement and concept recording instructions

Description of Analogous Patents Selection

The analogous patents utilized by the experimental analogy groups were found using the patent search methodology described in Chapter 4. The sub-functions used in each search were derived from the sub-functions required to fulfill the design problem functional requirements and constraints. The complete list of the design problem sub-functions is:

- Import
- Convert/Transform
- Transport
- Move/Rotate/Oscillate
- Collect
- Produce
- Export/Supply

where the functions are grouped by alternate functions which represent different flow domains. Acceptable solutions and analogies include any and/or all combinations of

alternate sub-functions. The specific sub-functions utilized in the patent searches for each analogy group are shown in Table 12.

Table 12: Functions searched for each analogy group

Group	Functionality Level of Analogies	Search Functions
Analogy Group 1	Single Sub-Function	Import
Analogy Group 2	Sub-Function Pair	Import, Convert
Analogy Group 3	All Sub-Functions	Import, Convert/Transform, Transport, Move/Rotate/Oscillate, Produce, Collect, Export/Supply

In order to minimize the time burden on the analogy groups, the searches for the analogous patents were completed prior to executing the experiment. Selections of four (4) patents were chosen from each set of search results based on both near-field and far-field analogies to the design problem as given in Table 13. The patents are selected from the search engine results based on both the relevancy score as well as the analogical distance as evaluated by a subject-matter expert.

Table 13: Analogous patents determined using Patent Search Tool

Group	Functionality Level of Analogies	Patent Title	Patent Number
Analogy Group 1	Single Sub-Function	Fuel injection apparatus having fuel pressurizing pump	5080079
		Inflating/deflating device for an inflatable air mattress	7571500
		Wireless communication device and signal receiving/transmitting method	7542009
		Paper guiding arrangement for a business machine	3567143
Analogy Group 2	Sub-Function Pair	Photovoltaic cell powered magnetic coil for operation of fluidic circuit flapper	3584636
		Virtual-wheeled vehicle	7588105
		Gray water interface valve systems and methods	7533426
		Air-blower tidal power generation device	7511386
Analogy Group 3	All Sub-Functions	Wave operated power apparatus	3603804
		System for recovering wasted energy from IC engine	7549412
		Method and device for capture, storage, and recirculation of heat energy	7549418
		Water current powered motor	7521816

During Phase 2 of the experiment, the analogy groups were presented the 4 patents corresponding to their respective group and given 10 minutes to study the patents (Chan, et al. 2011). They were given the patent abstract as well as a representative figure from

the patent. The textual description and pictorial descriptions were intentionally given together to mitigate the influence of representation (Linsey, et al., 2011). An example of an analogous patent as presented to the analogy group participants is shown in Figure 43.

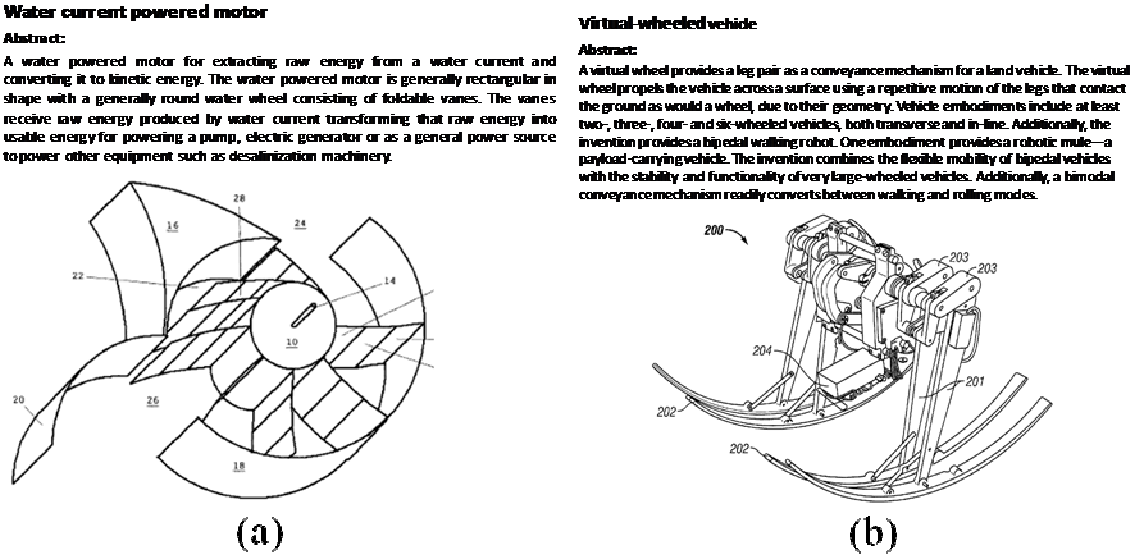


Figure 43: Example of analogous patent presented to analogy group participants: a) near-field analogy of power generator, b) far-field analogy of mechanism to import human energy

METRICS FOR EVALUATION

Goals of the concept generation process include the generation of as many unique ideas as possible, and the discovery of novel concepts within the theoretical space of ideas. A great breadth of potential solutions spanning as much of the design space as possible increases the potential for successfully determining the "best" solution per a given set of selection criteria (Otto and Wood, 2001, Ullman, 2003, Ulrich, 2004). Although a single concept can readily be identified as a comprehensive solution to the given design problem, determining what constitutes a single idea is more difficult to define. Previous literature has established rules or heuristics for defining what constitutes an independent idea (Shah, et al., 2000, Linsey, et al., 2005). Building on this knowledge

base, the definition of an independent idea utilized in this study is a physical embodiment that solves one of the sub-functions listed previously. Furthermore, the solution must consist of a *how* and *what* couple that satisfies the functional requirement of the corresponding sub-function as well as defines the solution *flow* domain per the Functional basis framework (Hirtz et al., 2002). The *how* specifies the component of the solution that acts upon the flow and the *what* defines the flow that is acted upon. For example, a solution for the flow independent function "Collect" would be "*air pressure* with *tank*" where the *how* is the "tank" collecting the "air pressure," and the air pressure is the *what* defining the specific flow domain as pneumatic potential energy. Following this definition scheme, the *Quantity of Ideas* metric is simply the sum total of unique ideas across all sub-functions for each participant.

The *Novelty of Ideas* metric was established as a measure of the rarity of a particular solution within each sub-function's design space. A complete design space for a particular function would be difficult to properly establish *a priori*, so an approximation was used which was defined as the initial set of solutions generated in Phase 1 for all participants. Novelty scores were computed for each sub-function solution using a formula utilized by Shah, et al. (2003) and Cagan, et al. (2011):

$$N_i = \frac{T_i - C_i}{T_i}$$

where T_i is the total number of unique solutions generated for sub-function i in Phase 1 across all participants, and C_i is the total number of solution tokens of the each solution in the first phase of ideation. The novelty score is a normalized value ranging from 0 to 1 for each idea. An example of the novelty scoring is given in Figure 44 for clarification.

	Sub-Function 1			
	Solution 1	Solution 2	Solution 3	Solution 4
Participant 1	●		●	●
Participant 2	●	●		
Participant 3	●		●	
Novelty Score (N_1)	$\frac{7-3}{7}=.57$	$\frac{7-1}{7}=.86$	$\frac{7-2}{7}=.71$	$\frac{7-1}{7}=.86$

Figure 44: Example of novelty scoring evaluation

Solutions generated in Phase 3 of the ideation process that did not occur in Phase 1 were given a novelty score of 1 since these concepts occurred outside the design space established prior to introduction of the patents in Phase 2. The final *Novelty of Ideas* score for each participant is the average of their sub-function novelty scores.

RESULTS

The experimental results for both the Quantity of Ideas and Novelty of Ideas are presented and discussed in the following sections. The statistical significance and implications of these results are reviewed with regard to the efficacy of the analogous patents on these metrics. The functionally level effects are reviewed to determine recommendations for analogy search strategies utilizing the Patent Analogy Search Tool. Figure 45 illustrates a typical result from an ideation phase. The complete experimental results are located in Appendix E.

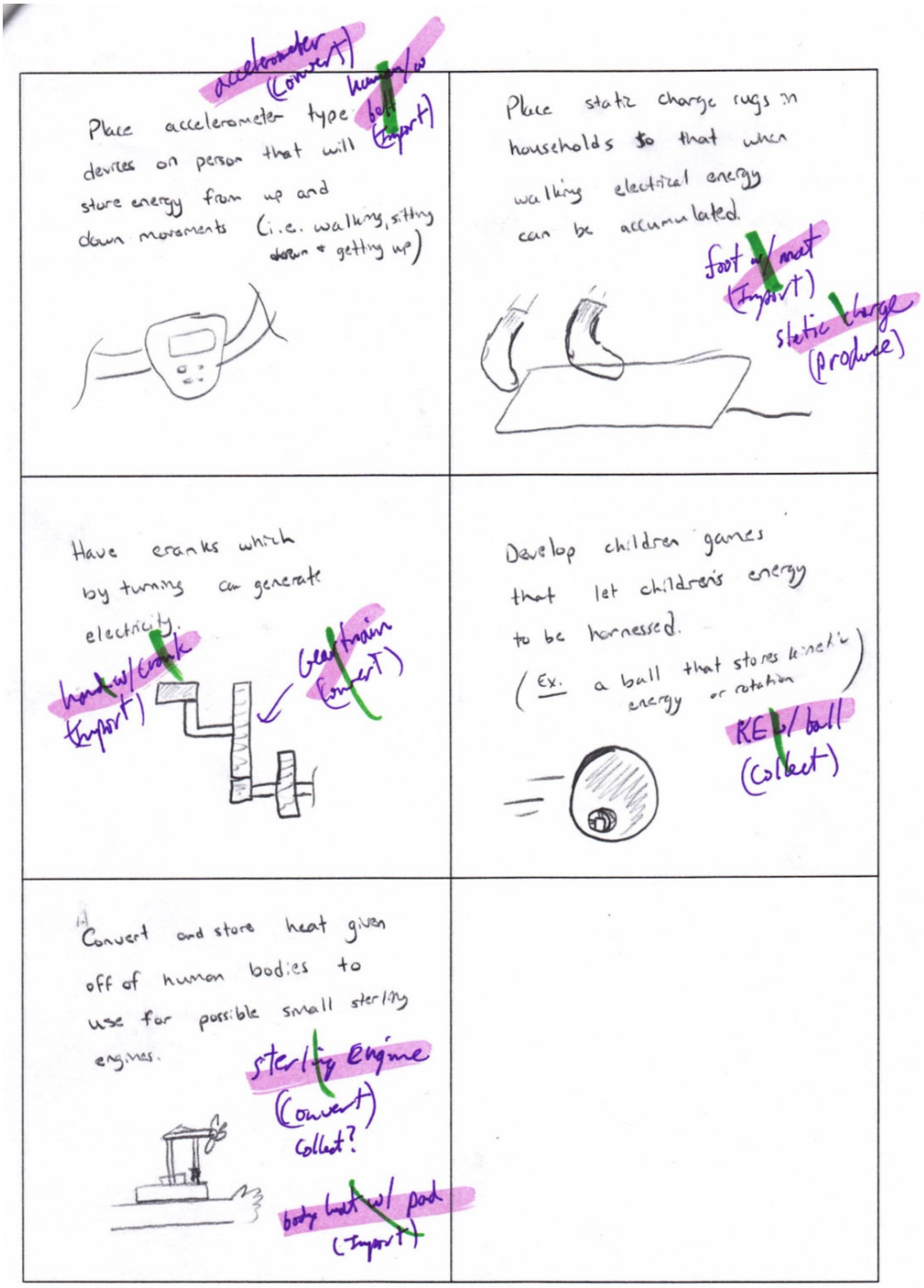


Figure 45: Example of ideation sheet with marked up ideas.

The quality of the sketches and descriptors was fairly consistent across all participants with few exceptions of both higher and poorer quality. The poor quality sketches were difficult to score due to unclear intent, therefore a conservative approach was taken for all scoring to ensure only explicit solutions were counted, not interpreted solutions. In Figure 46, three solutions are shown which are derived from analogies extracted from the patents given in Phase 2. In Figure 46(a), a novel solution for importing kinetic energy from humans is derived from a patent for importing wave motion. The floating bridge utilizes functional similarity to extract the energy from humans crossing the river. In Figure 46(b), a novel solution for converting energy is shown derived from the analogy of the wireless antenna. Wireless power transmission and conversion eliminates the need for local generation and storage, although many other feasibility issues such as signal attenuation limit the practicality of this solution. Finally in Figure 46(c), the patent solution is mapped between similar domains as a near-field analogy. The ocean wave power-harnessing solution is adapted to extract energy from the lower amplitude waves generated in river systems. These examples are typical of the novel solutions generated as a result of the introduction of analogous patent.

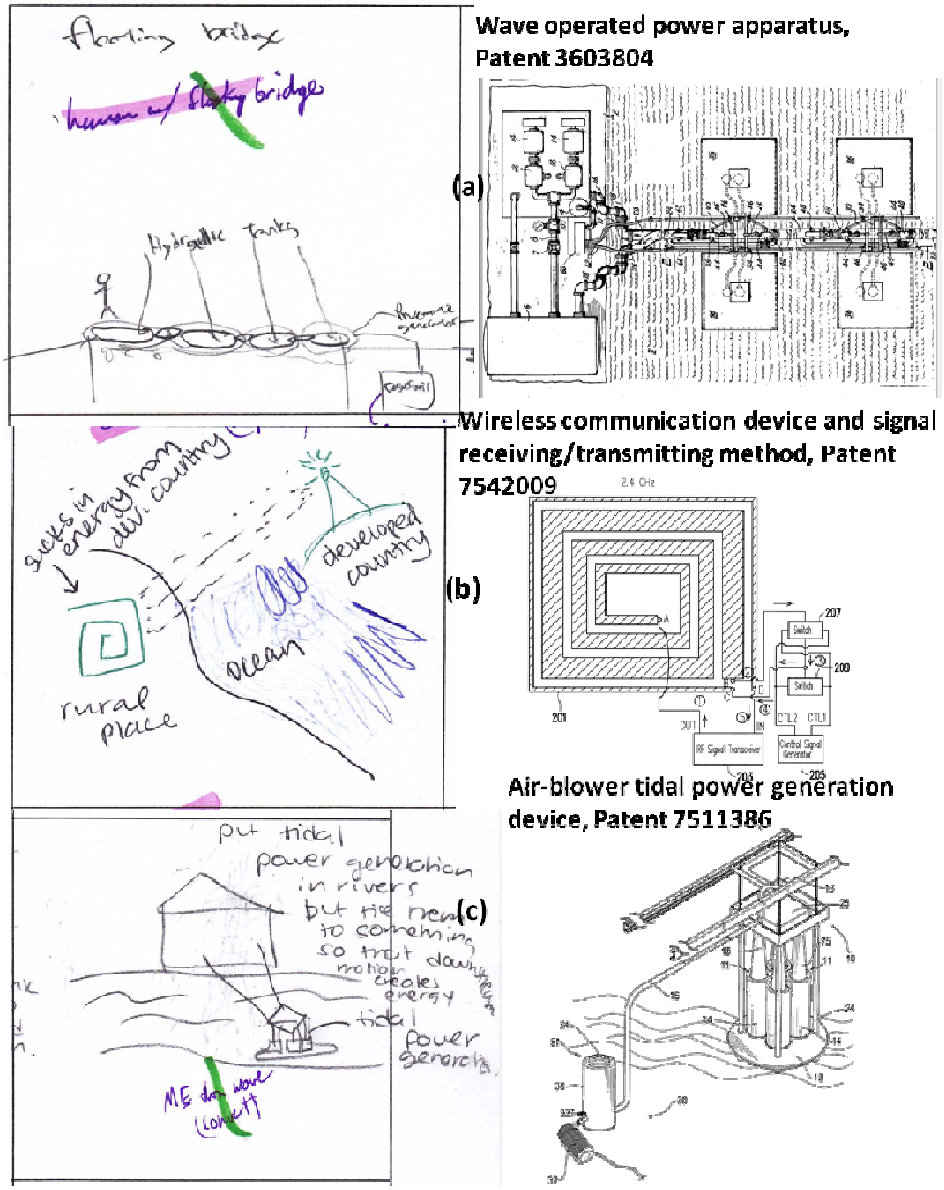


Figure 46: Sample solutions from each of the three analogy groups showing novel solutions derived from a patent: a) All functions group- floating bridge, b) Single function group- energy harvesting antenna, c) Function pair group- river power extraction device.

Quantity of Ideas

The average quantity of ideas for Phase 1 and Phase 3 combined was evaluated for each of the four experimental groups per the metric discussed in the previous section.

The results for each ideation phase and the total quantity of ideas are given in Figure 47. The overall high number of ideas generated by the participants can be attributed to previous training in ideation techniques through their design methodology courses.

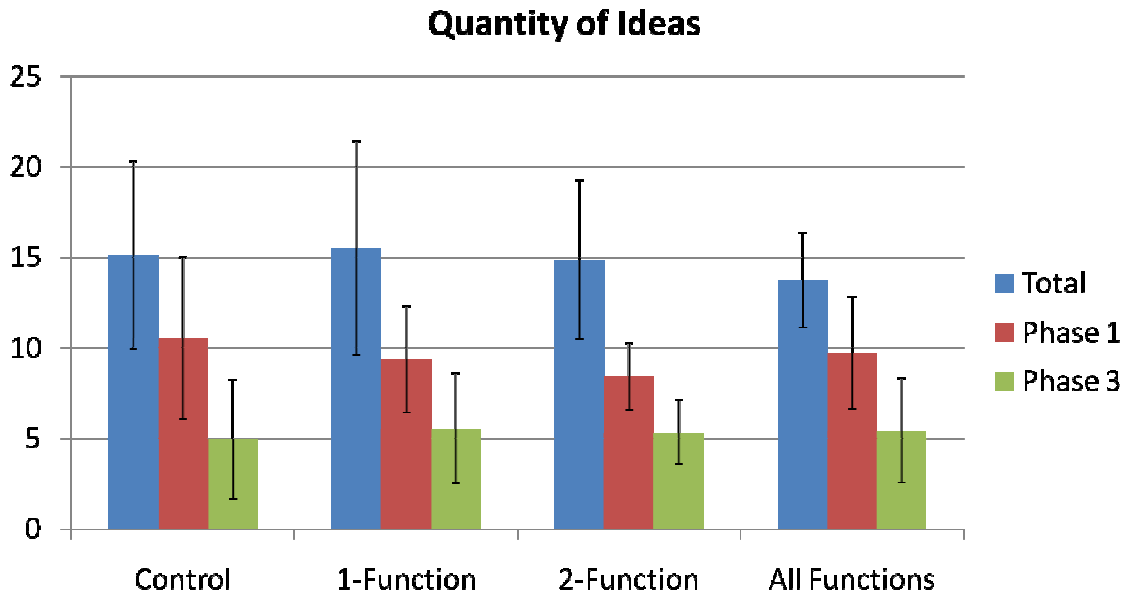


Figure 47: Average quantity of ideas generated for each group. Error bars show +/- 1 standard error.

There is a consistent falloff in the number of ideas generated from ideation Phases 1 and 3 across all groups. This result is in line with previous experimental data on ideation over time (Lindsey, 2007). The total quantity of ideas was also remarkably consistent across all groups. The Student's t-test for difference in means between the control group and the analogy groups in Table 14 shows that there is no statistically significant difference between the groups.

Table 14: Quantity of ideas Student's t-test results for each analogy group compared to control group.

	Control	*1-Function*	*2-Functions*	*All Functions*
Mean	15.12	15.47	14.88	13.76
Variance	26.99	34.39	19.11	6.94
P(T<=t)		0.427	0.444	0.174

This result implies that the patents had no positive or negative effect on the quantity of ideas generated. Although the analogous patents do not increase the quantity of ideas, they also do not have a negative impact such as reinforcing design fixation which would have a detrimental effect on the concept generation process.

Novelty of Ideas

The average novelty for each participant's ideas over the ideation Phases 1 and 3 was calculated per the novelty metric discussed in the previous section. The mean novelty and standard deviations for each experimental group are derived as shown in Figure 48.

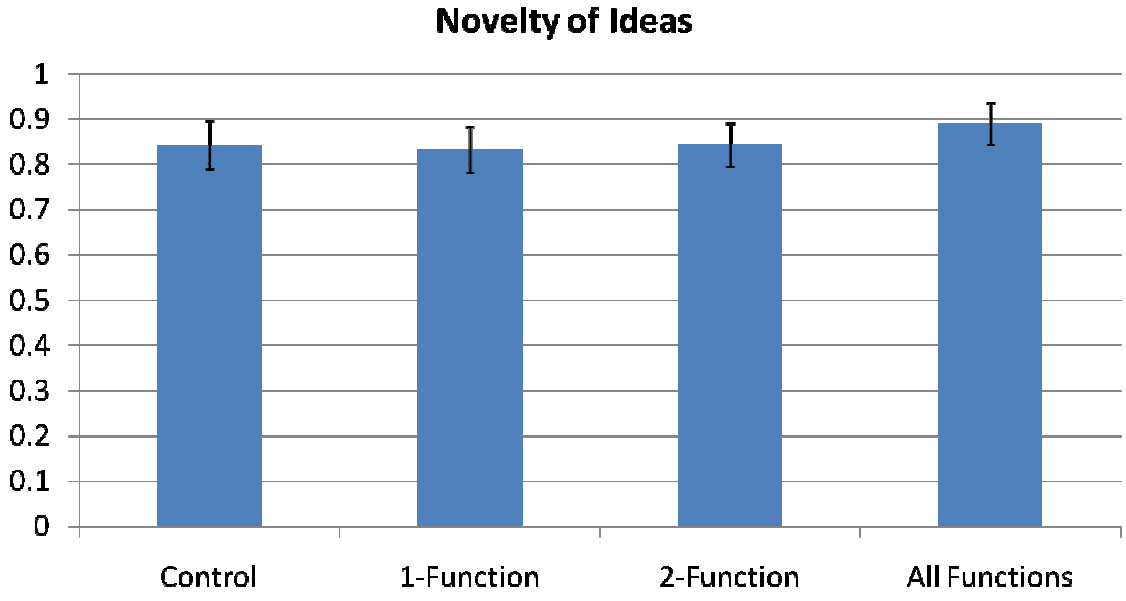


Figure 48: Average novelty of ideas generated for each group. Error bars show +/- 1 standard error.

Upon initial inspection, the All Functions group appears to have a larger mean and tighter distribution. The Student's t-test for difference in means was again used to determine whether a statistically significant difference in the mean group novelty score exists with respect to the control group. The results are given in Table 15.

Table 15: Novelty of ideas Student t-test results for each analogy group compared to control group

	Control	*1-Function*	*2-Functions*	*All Functions*
Mean	0.8416	0.8316	0.8429	0.8892
Variance	0.0029	0.0025	0.0023	0.0021
P(T<=t)		0.291	0.470	0.048

The All Function group does have a statistically significant higher average novelty than the control group at the 95% confidence level. The other analogy groups do not have a statistically significant difference in means. This result tends to confirm the analogous patents can improve the novelty of ideas generated during concept generation, but this effect appears to be dependent on the functionality level of the analogy. To confirm this insight, the t-test was performed on the mean novelty between the analogy groups by comparing both the 1-Function and 2-Funciton groups to the All Function group as shown in Table 16.

Table 16: Novelty of ideas Student t-test results within the analogy groups compared with All Functions group

	All Functions	*1-Function*	*2-Functions*
Mean	0.8892	0.8316	0.8429
Variance	0.0021	0.0025	0.0023
P(T<=t)		0.014	0.051

The All Function group has a statistically significant higher average novelty than the 2-Function group at the 94% confidence level, and a statistically significant higher average novelty than the 1-Function group at the 98% confidence level.

The strong significant effect on novelty due to the functionality level of the analogies was not expected. This effect could be attributed to a number of possibilities, including: 1) the narrow focus of the 1- and 2-function analogies causes design fixation within the constricted design space of those sub-functions or 2) the participants have a greater difficulty mapping the analogies at the narrowly focused functional level and the analogies are more apparent at the higher functional level. The design fixation cause is contradicted by the quantity of ideas result which showed the overall number of unique ideas to be the same across all groups. That contradiction lends support for the analogy mapping difficulty theory, but additional experiments would be required to verify the phenomenon.

CONCLUSIONS

The results of the experiment to test the patent analogy search tool efficacy for augmenting concept generation methods garnered several significant insights. The first insight was that analogical patents have no impact on the total quantity of unique ideas generated. The significance of this finding is that the introduction of analogous patent examples does not have a detrimental effect on concept generation through the phenomenon of design fixation.

The most important result supporting the efficacy of the patent analogy search tool was the significant effect of increased average novelty for the high functionally level analogy group. The All Functions group had a 5% higher average novelty rating than either the control group or the other analogy groups. This level of performance increase justifies the inclusion of the search tool into the concept generation process. Further experimentation should be conducted as part of the future work to identify the root cause of the functionality level effect. In the meantime, a high level, multi-function

representation of the design problem should be used for search query generation to obtain the best possible performance from the search tool. This finding is in agreement with the results of the case studies in Chapter 4 which concluded multiple searches over using multiple secondary functions maximizes the functional relevancy resolution. The next phase of experimental studies will investigate the performance of the Patent Analogy Search Engine compared to competing Design-by-Analogy methods such as the Wordnet procedure (Linsey, 2007).

Chapter 6: Conclusions and Future Work

CONCLUSIONS

Design-by-Analogy is an important tool for augmenting the Concept Generation process. Previously, few tools existed to systematically identify functional analogies for specific design problems across solution domains. The new Patent Analogy Search Engine and methodology provides a structured process and support tools to extract both near and far-field analogies from the vast design information contained in the USPTO patent database. The search engine has been developed based on knowledge gained from prior literature across multiple disciplines including functional modeling and representation, design repository methods, information retrieval theory and natural language processing. Using domain-independent representations and an expanded functional vocabulary, novel analogies can be extracted from the patent database to lead designers to novel solutions. The case study and experimental results both support the research hypothesis that a patent-based analogy search tool can identify non-obvious functional analogies and improve the novelty of concepts generated when utilized during ideation. The following sections summarize the conclusions regarding the search tool development and the lessons learned in applying the methodology to real-world case studies. The conclusions from the Patent Analogy Search experimental study are discussed. Finally, future work for the enhancement of the search tools and methodology is proposed

Patent-Based Analogy Search Tool Implementation

The Vector Space Model-based functional search engine provides domain-independent search capability within the patent database. The enabling technologies developed in this research are first the systematic derivation of a complete functional

vocabulary. The vocabulary is extracted directly from the target knowledge base of the USPTO patent database. Several natural language processing algorithms are developed to support the extraction of the relevant functional terms. The complete set of function verbs as defined using the convergence criteria in Chapter 3 are organized into a hierarchical structure modeled after the Functional Basis. The supplemental search tools such as the Query Generator interface and the Search Result Viewer interface were developed to enable effortless access to the design information contained in a limited sample of the patent database. The search engine development leads to insights regarding the relationship between functionally rich patents and analogy mapping difficulties due to the excessively broad functional coverage. Several potential enhancements to the search engine were also identified and are discussed future work.

Patent Analogy Search Methodology and Case Studies

In Chapter 4, the systematic method for searching for analogies in the patent database was described. The functional model for a given design problem is simplified and mapped to a domain-independent semantic representation expressed in the expanded Functional Basis vocabulary. Using this approach, the Patent Analogy Search Engine methodology successfully reproduced the results of the well known Design-by-Analogy guitar pickup winder. Even given the limited patent coverage in the prototype patent database, the search process extracted three of the five top analogous products for the pickup winder. The key insight gained from the pickup winder case study was generating large query vectors by searching over multiple Primary and Secondary functions detrimentally reduces the total relevancy score metric resolution. The reduction in resolution requires the user to search many more patents to extract the desired patent analogies. Going forward, the search approach utilized multiple searches over fewer

functions. Using lessons learned from the pickup winder case study, the automated window washer design problem used multiple search queries generated with fewer Secondary functions. Six individual searches were performed and the identified analogies included both near and far-field analogies. The far-field analogies were extracted for coupling the device to vertical surfaces and for removing debris using cryogenic fluids as well as a very near-field analogy exactly describing an automated window washing device. The methodology developed and verified by case study was then used to extract the analogous patents for the concept generation experiment.

Insights from Concept Generation Experiment

The experimental study to test the patent analogy search tool efficacy supports the hypothesis that analogous patents increase the overall novelty of concepts generation during ideation. In the experiment described in Chapter 5, the All Functions group had a 5% higher average novelty rating than either the control group or the other analogy groups. The significant performance increase justifies the inclusion of the search tool into the concept generation process. The inclusion of analogous patents in the ideation process had no significant effect on the quantity of ideas generated. The lack of effect on quantity of ideas secondarily verifies that the introduction of patent examples does not propagate the phenomenon of design fixation to detrimental effect on concept generation. The underlying principle for increased average novelty for the high functionally level analogy group is not completely understood from this initial study. Further experimentation should be conducted as part of the future work to identify the root cause of the functionality level effect. Additionally, verification of the experimental result should be conducted using a secondary rater to determine the inter-rater

FUTURE WORK

Search Engine Extensions and Enhancements

The Patent Analogy Search Engine has been shown to be effective for identifying functional analogies from the patent database and for increasing the novelty of concepts generated during ideation. Although, the initial results are promising continued improvements to the search engine design could further enhance the tool's efficacy. The first improvement proposed is simply increasing the patent database coverage. The current prototype implementation is near the limit of the tool used for development. The MATLAB program provides rapid prototyping capabilities, but is not well suited for supporting large databases. Porting the search functionality to a dedicated database language such as MySQL would enable complete indexing and storage of the USPTO patent database.

Further research is needed to optimize the *total relevancy score* metric. Including patent length normalization into the Patent Functional Content metric could be used to minimize the bias resulting from longer patents including a broader range of function terms. Additionally, a rigorous experimental study to determine the optimal relevancy score coefficient weights must be conducted to verify the results from the parametric process utilized in this research.

Additional extensions to the search engine to be investigated are the inclusion of customer needs utilizing system attribute terms as adjectives. The attribute terms would be implemented as context limiters used to augment the similarity metric. Some examples of attribute term adjectives are *quickly*, *cheap*, *light*, etc.

Functionality Level Effect Verification

Although the introduction of analogies during ideation increased the overall solution novelty for the All-functions group, the novelty gap between the All-functions analogy group and the other analogy groups in the experimental study is not well understood. The single function analogy and function-couple analogy groups showed no statistical improvement over the control group. A follow up experimental study should be conducted to specifically test for this effect. The insight gained if a significant effect correlated with functionality level is identified would greatly impact the search methodology with regard to query generation. If multiple, high level functions consistently produce higher solution novelty, the total relevancy score metric would require modification to improve the poor similarity score resolution for large query vectors.

A secondary test of the functionality level effect would be modifying the structure of the extending Functional Basis. An experimental study to evaluate the effect of combining and separating the Secondary function groups and Correspondent functions would be used to determine the optimal number of Secondary function categories. Fewer Secondary functions equates to larger query vectors due to the increased number of Correspondents. If fewer Secondary functions produce higher solution novelty, the total relevancy score metric would require similar modification to improve the similarity score resolution.

Appendix A: Boolean Model Patent Search Matlab Code

MAIN FUNCTION

```
function [patclass_hit,queryterms]=runsearch()
%Main program- loads all other programs
clear
clc
% prompt = {'Choose Patent Class Range or Choose 0 for all'};
% title = 'Analogy Search';
% lines = 1;
% def = {'10'};
% answer = inputdlg(prompt,title,lines,def,'on');
% range_class = str2num(answer{1});

range_class = 0;
[patclass_hit,queryterms]= analogysearch(range_class);
disp('Search is finished.')
if isempty(patclass_hit(1).num)
    test = 1;
    while test
        prompt = {'No results found. Use different search terms?'};
        title = 'Search Again';
        lines = 1;
        def = {'Yes'};
        answer = inputdlg(prompt,title,lines,def,'on');
        test = strcmp(lower(answer{1}),'yes');
        if test
            [patclass_hit,queryterms]= analogysearch(range_class);
        end
    end
else
    str_f = gen_filename(queryterms,1);
    filestr = ['File will be saved in Results directory as ',str_f];
    disp(filestr)
    file_mat = strcat(gen_filename(queryterms,0),'.mat');
    save(file_mat);
    excel_write(patclass_hit,queryterms);
    searchviewer(patclass_hit,queryterms);
end
```

SEARCH FUNCTION

```
function [patclass_hit,queryterms]= analogysearch(range_class)
%[patclass_hit,queryterms]=analogysearch(range_class);
%
%RANGE_CLASS variable is the number of patent classes to search.
%Option for RANGE_CLASS is a number [1 to 400] or 'all' to search all
classes.
%
```

```
%Copy/paste function above and replace 'RANGE_CLASS' with desired value
to run
%the analogy search.
%
%For multiple Functions, System Inputs and Customer Needs entries use
comma
%and space to separate terms, e.g. vibrate, branch, measure.
%
%For Customer Needs phrases, use quotes to enclose phrase, e.g. "Low
Cost"
%
%Setting Limit Verb Tense to 'Yes' limits search terms to 'ing' and
'ed'
%tenses, e.g. 'vibrating' and 'vibrated'. Useful when most common usage
of
%term is not in a functional sense.
%
%Search field options are 'Title', 'Abstract', 'Description', and 'All'
and
%controls in which section of the patent the query occurs
%
%See also EXCEL_OUTPUT SEARCHPAGEVIEWER
%range_class = 10;
if ischar(range_class)
    error('Bad value of RANGE_CLASS')
elseif range_class<0||range_class>400
    error('Bad value of RANGE_CLASS')
end
prompt = {'Enter Function e.g. separate, channel, etc.',...
    'Limit Verb Tense?','Enter System Inputs e.g. fluid, spool,
etc',...
    'Enter Customer Needs and Context Limiters e.g. "low cost", robust,
etc.',...
    'Search Field'};%,'Web Browser Display'};
title = 'Analogy Search Query Generator';
lines = 1;
def = {'','No','','','Abstract'};%,'off'};
options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='none';
answer = inputdlg(prompt,title,lines,def,options);
functions = lower(answer{1});
queryterms.limit = lower(answer{2});
flows = lower(answer{3});
cns = lower(answer{4});
queryterms.field = lower(answer{5});
%queryterms.web = lower(answer{6});
matches = 0;
patclassfile = 'patent_classes.txt';
patclass = loadpatclass(patclassfile);
len_class = length(patclass.num);
%parse function and flow query
```

```
[queryterms.funct,queryterms.flow,queryterms.cns] =
parse_query(functions,flows,cns);
if ~strcmp(queryterms.flow,'')
    len_flow=length(queryterms.flow);
else
    len_flow = 0;
end

if ~strcmp(queryterms.cns,'')
    len_cn=length(queryterms.cns);
else
    len_cn = 0;
end

num_terms = [length(queryterms.funct) len_flow len_cn];

k = 0;
nummatch_temp =[];
if range_class == 0
    len_class = len_class;
else
    len_class = range_class;
end
for i = 1:len_class
    [url,searchurl] =
searchstring_v2(queryterms,num_terms,num2str(patclass.num(i)));
    disp(strcat('Searching:',patclass.title(i)))
    matches = find_num_matches(url);
    if  matches>0
        k= k+1;
        num_temp(k) = patclass.num(i);
        title_temp(k) = patclass.title(i);
        url_temp{k} = searchurl;
        nummatch_temp(k) = matches;
    end
end
if ~isempty(nummatch_temp)
    [b,ind] = sort(nummatch_temp);
    len = length(b);
    j = len;
    for i = 1:len
        patclass_hit(i).nummatch = b(j);
        ind_temp = ind(j);
        patclass_hit(i).num = num_temp(ind_temp);
        patclass_hit(i).title = title_temp(ind_temp);
        patclass_hit(i).url = url_temp(ind_temp);
        j = j-1;
    end
else
    patclass_hit(1).nummatch = [];
    patclass_hit(1).num = [];
    patclass_hit(1).title = [];
```

```
    patclass_hit(1).url = [];
end
patclass_hit;
queryterms;
```

QUERY GENERATOR

```
function [url,search] = searchstring_v2(queryterms,nums,class)

num_funct = nums(1);
num_flow = nums(2);
num_cns = nums(3);
queryfunct = '';
queryflow = '';
querycns = '';

%Generate Function Query
if num_funct~=0
    for i = 1:num_funct
        f1_inv = inv_stemmer(queryterms.funct(i));
        if i==1
            if ~strcmp(queryterms.limit,'yes')
               queryfunct =
strcat('%28',f1_inv(1),'+OR+',f1_inv(2),'+OR+',f1_inv(3),'+OR+',f1_inv(
4),'%29');
            else
               queryfunct =
strcat('%28',f1_inv(3),'+OR+',f1_inv(4),'%29');
            end
        else
            if ~strcmp(queryterms.limit,'yes')
                queryfunct =
strcat(queryfunct,'+AND+%28',f1_inv(1),'+OR+',f1_inv(2),'+OR+',f1_inv(3
),'+OR+',f1_inv(4),'%29');
            else
                queryfunct =
strcat(queryfunct,'+AND+%28',f1_inv(3),'+OR+',f1_inv(4),'%29');
            end
        end
    end
end

%Generate Flow Query
if num_flow~=0
    for i = 1:num_flow
        f2_inv = pluralize(queryterms.flow(i));
        if i == 1
            queryflow = strcat(f2_inv(1),'+OR+',f2_inv(2));
        else
            queryflow =
strcat(queryflow,'+OR+',f2_inv(1),'+OR+',f2_inv(2));
        end
```

```
    end
end

%Generate CN query
if num_cns~=0
    for i = 1:num_cns
        if i == 1
            k = strfind(char(queryterms.cns(i)), '+');
            if ~isempty(k)
                querycns = strcat('%22',queryterms.cns(i),'%22');
            else
                querycns = queryterms.cns(i);
            end
        else
            k = strfind(char(queryterms.cns(i)), '+');
            if ~isempty(k)
%                querycns
=strcat(querycns,'+AND+%22',queryterms.cns(i),'%22');
                querycns =
strcat(querycns,'+OR+%22',queryterms.cns(i),'%22');
            else
%                querycns = strcat(querycns,'+AND+',queryterms.cns(i));
                querycns = strcat(querycns,'+OR+',queryterms.cns(i));
            end
        end
    end
end

%funct only
if ~strcmp(queryfunct,'')&&strcmp(queryflow,'')&&strcmp(querycns,'')
    query = queryfunct;
%flow only
elseif
strcmp(queryfunct,'')&&~strcmp(queryflow,'')&&strcmp(querycns,'')
    query = queryflow;
%cn only
elseif
strcmp(queryfunct,'')&&strcmp(queryflow,'')&&~strcmp(querycns,'')
    query = querycns;
%funct+flow
elseif
~strcmp(queryfunct,'')&&~strcmp(queryflow,'')&&strcmp(querycns,'')
    query = strcat(queryfunct,'+AND+%28',queryflow,'%29');
%funct+cn
elseif
~strcmp(queryfunct,'')&&strcmp(queryflow,'')&&~strcmp(querycns,'')
    query = strcat(queryfunct,'+AND+%28',querycns,'%29');
%flow+cn
elseif
strcmp(queryfunct,'')&&~strcmp(queryflow,'')&&~strcmp(querycns,'')
    query = strcat(queryflow,'+AND+%28',querycns,'%29');
%funct+flow+cn
```

```
else
    query =
strcat(queryfunct,'+AND+%28',queryflow,'%29+AND+%28',querycns,'%29');
end

switch lower(char(queryterms.field))
    case 'all'
        query = query;
    case 'title'
        query = strcat('ttl%2F%28',query,'%29');
    case 'abstract'
        query = strcat('abst%2F%28',query,'%29');
    case 'description'
        query = strcat('spec%2F%28',query,'%29');
    otherwise
        disp('Bad search field')
        return;
end
%add class to search
query = strcat('ccl%2F',class,'+AND+',query);

% US patents only
 search =
char(strcat('http://www.freepatentsonline.com/result.html?edit_alert=&s
rch=xprtsrch&query_txt=',query,...

'&uspat=on','&date_range=all','&stemming=off&sort=relevance&search=Sear
ch'));
try
     url = urlread(search);
 catch
     url = '';
 end
```

SEARCH VIEWER FUNCTION

```
function varargout = searchviewer(varargin)
% SEARCHVIEWER M-file for searchviewer.fig
%      SEARCHVIEWER, by itself, creates a new SEARCHVIEWER or raises
the existing
%      singleton*.
%
%      H = SEARCHVIEWER returns the handle to a new SEARCHVIEWER or the
handle to
%      the existing singleton*.
%
%      SEARCHVIEWER('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in SEARCHVIEWER.M with the given input
arguments.
%
```

```
%      SEARCHVIEWER('Property','Value',...) creates a new SEARCHVIEWER
or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before searchviewer_OpeningFunction gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to searchviewer_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help searchviewer

% Last Modified by GUIDE v2.5 20-Jun-2007 02:56:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @searchviewer_OpeningFcn, ...
    'gui_OutputFcn',  @searchviewer_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before searchviewer is made visible.
function searchviewer_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
guidata(hObject, handles);
load_search_data(varargin,handles);
% UIWAIT makes searchviewer wait for user response (see UIRESUME)
% uiwait(handles.searchviewer);
```

```
function load_search_data(varargin,handles)

if isempty(varargin)
    uiload
elseif length(varargin) == 1&&(2 == exist(varargin{1},'file'))
    filename = varargin{1};
    load(filename)
elseif length(varargin) == 2 %&&
strcmpi(varargin{1},'patclass_hit')&&strcmpi(varargin{2},'queryterms')
    patclass_hit = varargin{1};
    queryterms = varargin{2};
else
    errordlg('Patent Search Data not Found')
    delete(handles.searchviewer);
    return
end
handles.patclass = patclass_hit;
handles.query = queryterms;
handles.Index = 1;
Current_terms
=strvcat(char(queryterms.funct),char(queryterms.flow),char(queryterms.c
ns));
guidata(handles.searchviewer,handles)
set(handles.num_pats,'string',handles.patclass(1).nummatch)
set(handles.search_term,'string',Current_terms)
set(handles.pat_class,'string',handles.patclass(1).title)
set(handles.index_num,'string',handles.Index)
set(handles.tot_num,'string',length(handles.patclass))

% --- Outputs from this function are returned to the command line.
function varargout = searchviewer_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;


function varargout = prev_next_Callback(h, eventdata, handles, str)
% Get the index_num pointer and the addresses
index = handles.Index;
patclasses = handles.patclass;
queryterm = handles.query;
% Depending on whether Prev or Next was clicked change the display
switch str
    case 'Prev'
        i = index - 1;
        if i < 1
            i = length(patclasses);
        end
    case 'Next'
        i = index + 1;
        if i > length(patclasses)
            i = 1;
        end
```

```
end
% Get the appropriate data for the index_num in selected
handles.Index = i;
Current_class = patclasses(i).title;
Current_num = patclasses(i).nummatch;
set(handles.num_pats,'string',Current_num)
set(handles.pat_class,'string',Current_class)
set(handles.index_num,'string',handles.Index)
%Current_terms =strvcat(queryterm.funct,queryterm.flow,queryterm.cns);
%set(handles.search_term,'string',Current_terms)

guidata(h,handles)

% --- Executes on button press in open_browser.
function open_browser_Callback(hObject, eventdata, handles)
% hObject    handle to open_browser (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
index = handles.Index;
patclasses = handles.patclass;
srch_url = patclasses(index).url;
web(char(srch_url),'-browser');
```

Appendix B: Vector Space Model Source Code

FAST PATENT INDEXER FUNCTIONS

```
function varargout = indexer_gui_v2(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@indexer_gui_v2_OpeningFcn, ...
                   'gui_OutputFcn',
@indexer_gui_v2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before indexer_gui_v2 is made visible.
function indexer_gui_v2_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to indexer_gui_v2 (see
VARARGIN)

% Choose default command line output for indexer_gui_v2
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject,handles);
% load master_terms_list
% handles.master_list = master_terms;
get_last_pat_indexed(hObject,handles,[7600 1715]);

% UIWAIT makes indexer_gui_v2 wait for user response (see
UIRESUME)
% uiwait(handles.indexer_gui_v2);


% % --- Outputs from this function are returned to the
command line.
 function varargout = indexer_gui_v2_OutputFcn(hObject,
eventdata, handles)
     varargout{1} = handles.output;
%
%
%

function current_term_Callback(hObject, eventdata, handles)
% hObject    handle to current_term (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
new_term = get(hObject,'string');
handles.term(handles.term_track) = new_term;
position =
strcat(dec2base27(handles.term_pos),num2str(handles.pat_tra
ck));
Select(Range(handles.excel,position));
set(handles.excel.Selection,'Value',new_term);
guidata(hObject,handles);
%
% % Hints: get(hObject,'String') returns contents of
current_term as text
% %        str2double(get(hObject,'String')) returns
contents of current_term as a double
% % --- Executes during object creation, after setting all
properties.
function current_term_CreateFcn(hObject, eventdata,
handles)
% Hint: edit controls usually have a white background on
Windows.
```

```
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgr
oundColor'));
end

%
% % --- Executes on button press in finished.
function finished_Callback(hObject, eventdata, handles)
% hObject    handle to finished (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

invoke(handles.excel.ActiveWorkbook,'Save');
handles.excel.Workbooks.Close;
handles.excel.Quit;
handles.excel.delete;
close;

 %get value of last patent indexed
function get_last_pat_indexed(hObject,handles,pat_range)
% file = strcat(pwd,'\patents_thesaurus_v3.xls');
file = strcat(pwd,'\patents_tagger_parsed.xls');
handles.excel = actxserver('Excel.Application');
handles.excel.Workbooks.Open(file);
handles.excel.Visible = 1;
handles.pat_track = 1;
handles.end_pat = 50000;
guidata(hObject,handles);
get_terms(hObject,handles)

% %%%%%%%%%%%%%
% %%%%%%%%%%
function get_terms(hObject,handles)
indx_pos = handles.pat_track;
terms = 0;
handles.term = {''};
handles.term_color = 6;
while ~terms
```

```
    position = strcat('B',num2str(indx_pos));
    Select(Range(handles.excel,position));
    pat_num = get(handles.excel.Selection,'Value');
    set(handles.pat_num,'string',num2str(pat_num));
    position = strcat('A',num2str(indx_pos));
    Select(Range(handles.excel,position));
    handles.excel.Selection.Interior.ColorIndex = 8;
    check = 0;
    count = 4;
    while check == 0
        position =
strcat(dec2base27(count),num2str(indx_pos));
        Select(Range(handles.excel,position));
        word = get(handles.excel.Selection,'Value');
        if isnan(word)
            check = 1;
            if count > 4
                terms = 1;
            else
                terms = 0;
            end
        else
            handles.term{count-3} = word;
        end
        count = count +1;
    end
    indx_pos = indx_pos+1;
end
handles.pat_track = (indx_pos-1);
handles.term_track = 1;
handles.term_pos = 4;
set(handles.finished,'Visible','on')
guidata(handles.indexer_gui_v2,handles);
set(handles.current_term,'string',handles.term(1))



function set_type_Callback(hObject, eventdata, handles,str)
switch str
    case 'funct_term'
        handles.term_color = 6;
    case 'delete_term'
        handles.term_color = 1;
    case 'sysin_term'
```

```
        handles.term_color = 3;
    case 'attrib_term'
        handles.term_color = 4;
    case 'component_term'
        handles.term_color = 5;
end
position =
strcat(dec2base27(handles.term_pos),num2str(handles.pat_tra
ck));
Select(Range(handles.excel,position));
handles.excel.Selection.Interior.ColorIndex =
handles.term_color;
handles.term_pos = handles.term_pos+1;
handles.term_track = handles.term_track+1;
set(handles.finished,'Visible','off');
guidata(hObject, handles);
next_term(hObject,handles);


function next_term(hObject,handles)
term = handles.term_track;
len_term = length(handles.term);
if term <= len_term
    set(handles.current_term,'string',handles.term(term))
    guidata(hObject,handles);
else
    handles.pat_track = handles.pat_track+1;
    guidata(hObject,handles);
    get_terms(hObject,handles);
end


% --- Executes on button press in back.
function back_Callback(hObject, eventdata, handles)
% hObject    handle to back (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
handles.term_pos = handles.term_pos-1;
handles.term_track = handles.term_track-1;
position =
strcat(dec2base27(handles.term_pos),num2str(handles.pat_tra
ck));
```

```
Select(Range(handles.excel,position));
handles.excel.Selection.Interior.ColorIndex =
handles.term_color;
term = handles.term_track;
set(handles.current_term,'string',handles.term(term))
guidata(hObject,handles);


function find_def_Callback(hObject, eventdata, handles)
% hObject    handle to open_browser (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
word = handles.term(handles.term_track)
srch_url =
strcat('http://www.google.com/search?q=define%3A+',word);
web(char(srch_url),'-browser');


function next_Callback(hObject, eventdata, handles)
handles.term_pos = handles.term_pos+1;
handles.term_track = handles.term_track+1;
set(handles.finished,'Visible','off');
guidata(hObject, handles);
next_term(hObject,handles);
```

SUFFIX STRIPPER FUNCTION

```
function stem = suffix_stripper(str)

check = 0;
if ~isempty(regexp(str,'(sses)\>','start'))
    str = regexprep(str,'(sses)\>','ss');
elseif ~isempty(regexp(str,'(ies)\>','start'))
    str =regexprep(str,'(ies)\>','y');
elseif ~isempty(regexp(str,'[^seui](s)\>','start'))
    str =str(1:length(str)-1);
elseif ~isempty(regexp(str,'[^aiou](es)\>','start'))
    str =str(1:length(str)-2);
    check = 1;
end

str = regexprep(str,'(?<=y)(ed)\>','');
```

```
if ~isempty(regexp(str,'(eed)\>','start'))
elseif~isempty(regexp(str,'(?<=[aeiou]+\w*)(ed)\>','start')
)
    str = regexprep(str,'(?<=[aeiou]+\w*)(ed)\>','');
    check = 1;
end

str = regexprep(str,'(?<=y)(ing)\>','');
if ~isempty(regexp(str,'(?<=[aeiouy]+\w*)(ing)\>','start'))
    str = regexprep(str,'(?<=[aeiouy]+\w*)(ing)\>','');
    check = 1;
end

if check
    if
~isempty(regexp(str,'([aeiou][^aeiouv][eiou][t])\>','start'
))
        str = strcat(str,'e');
    elseif
~isempty(regexp(str,'([^aeiou][^aeiou][eiou][t])\>','start'
))
    elseif
~isempty(regexp(str,'[^aeiou][aiou][^aeiouwxy]\>','start'))
...

&&isempty(regexp(str,'[aeiou]\w*(or)\>','start'))
        str = strcat(str,'e');
    elseif ~isempty(regexp(str,'([^slfe])\1\>','tokens'))
        str = regexprep(str,'([^s])\1\>','$1');
    elseif ~isempty(regexp(str,'[^aeiou](i)\>','start'))
        str =regexprep(str,'(i)\>','y');
    elseif
~isempty(regexp(str,'(ang)\>|([ou]r[sgc])\>|(abl)\>|(icl)\>
|([^aeiou]at)\>|([aeiou]dg)\>|(pl)\>|(dg)','start'))
        str = strcat(str,'e');
    elseif
~isempty(regexp(str,'([yiea]z)\>|([aui]r)\>|(eas)\>|v\>|(yc
l)\>|(ers)\>','start'))
        str = strcat(str,'e');
    elseif
~isempty(regexp(str,'(ais)\>|([iu]at)\>|(br)\>|([mn][gdb]l)
\>','start'))
        str = strcat(str,'e');
    end
```

```
    if
~isempty(regexp(str,'([io][ms]ite)\>|(sone)\>|(eare)\>|(ivo
te)\>','start'))
        str = regexprep(str,'(e)\>','');
    elseif ~isempty(regexp(str,'(hous)\>','start'))
        str = strcat(str,'e');
    elseif
~isempty(regexp(str,'([aie][rpn][gsc])\>|(guid)\>|(spher)\>
|(caus)\>|([^i]*cit)\>','start'))
        str = strcat(str,'e');
    end
end

stem = str;
```

PREFIX STRIPPER FUNCTION

```
function [root,flag,hit] =
prefix_stripper(list,str,prefixes)
% Prefix stripper: sub, re, un, de, under, mis, over, pre,
post, non, counter, out, inter, micro, up, super, en, co,
dis, hyper, ultra
len = length(prefixes);
flag(len) = 0;
for i = 1:len
    expr = strcat('\<(',prefixes{i},')');
    if ~isempty(regexp(str,expr,'start'))
        temp_str = regexprep(str,expr,'');
        check = check_root(list,temp_str);
        if check == 0
            flag(i) = 1;
            root = temp_str;
            hit = 1;
            return;
        else
            hit = 0;
            root = str;
        end
    else
        hit = 0;
        root = str;
    end
end
```

PARSING FUNCTION

```
function [title,abstract,class,claims,description] =
parse_pat_all(patent)


[m s e] = regexp(patent,'Title:.+?(</b>){1}',
'match','start', 'end');
m2 = regexprep(m,'<.*?>|[\f\n\r\t\v]','');
title = regexprep(m2,'Title:\s*','');
if ~isempty(m)
    patent = patent(e:length(patent));
end

[m s e] = regexp(patent,'Abstract:.+?(</div>){1}',
'match','start', 'end');
m2 = regexprep(m,'<.*?>|[\f\n\r\t\v]','');
abstract = regexprep(m2,'Abstract:\s*','');
if ~isempty(m)
    patent = patent(e:length(patent));
end

[m s e] = regexp(patent,'Primary Class:.+?(</div>){1}',
'match','start', 'end');
m2 = regexprep(m,'<.*?>|[\f\n\r\t\v]','');
class = regexprep(m2,'Primary Class:\s*|\s*','');
if ~isempty(m)
    patent = patent(e:length(patent));
end

[m s e] = regexp(patent,'Claims:.+?(</div>){1}',
'match','start', 'end');
m2 = regexprep(m,'<.*?>|[\f\n\r\t\v]','');
claims = regexprep(m2,'Claims:\s*','');
if ~isempty(m)
    patent = patent(e:length(patent));
end

[m s e] = regexp(patent,'Description:.+?(</div>){1}',
'match','start', 'end');
m2 = regexprep(m,'<.*?>|[\f\n\r\t\v]','');
description = regexprep(m2,'Description:\s*','');
%patent = patent(e:length(patent));
```

TERM EXTRACTION FUNCTION

```
disp('Count:');disp(1);
pat_file = strcat(num2str(patent_list(1)),'_all.txt');
input_file = strcat(pwd,'\Patent_parsed_again\',pat_file);
fid = fopen(input_file,'r');
txt=fscanf(fid,'%c');
fclose(fid);


[a,b]=regexp(txt,'end_title|end_abstract|end_class|end_clai
ms|end_description','match','split');

%patent.number = pat_num;
patent.title = b(1);
patent.abstract = b(2);
patent.class = b(3);
patent.claims = b(4);
patent.description = b(5);

fid = fopen('temp_in.txt','w');
fprintf(fid,'%s',b{2})
fclose(fid);
output_file = strcat('temp_out.txt');
dos_str = ['tag-english ','temp_in.txt',' ',output_file];
dos(dos_str);
[a,b]= tag_reader(output_file,stop_words);
functions = a;
attributes =b;
```

TERM TAG READER

```
function [verbs,descriptors] =
tag_reader(filename,stop_words)

str= textread(filename,'%s','delimiter','\n');
k=1;
verb =  {''};
adjective = {''};
adverb = {''};

%Extract verbs
for i = 1:length(str)
```

```
    [a,b]=regexp(char(str(i)),'VV[A-Z]?','match','split');
    if ~isempty(a)
        verb_temp =regexp(char(b(2)),'\w*\w','match');
        if length(verb_temp)>1;verb_temp = 'a';end
        if check_term(verb,verb_temp)
            if check_term(stop_words,verb_temp)
                verb(k)=verb_temp;
                k=k+1;
            end
        end
    end
end
%Extract adjectives
k=1;
for i = 1:length(str)
    [a,b]=regexp(char(str(i)),'JJ[A-Z]?','match','split');
    if ~isempty(a)
        adjective_temp =regexp(char(b(2)),'\w*\w','match');
        if
length(adjective_temp)>1;adjective_temp=adjective_temp(leng
th(adjective_temp));end;
        if check_term(adjective,adjective_temp)
            if check_term(stop_words,adjective_temp)
                adjective(k)=adjective_temp;
                k=k+1;
            end
        end
    end
end
%Extract adverbs
k=1;
for i = 1:length(str)
    [a,b]=regexp(char(str(i)),'R[BP][A-
Z]?','match','split');
    if ~isempty(a)
        adverb_temp =regexp(char(b(2)),'\w*\w','match');
        if
length(adverb_temp)>1;adverb_temp=adverb_temp(length(adverb
_temp));end;
        if check_term(adverb,adverb_temp)
            if check_term(stop_words,adverb_temp)
                adverb(k)=adverb_temp;
                k=k+1;
            end
```

```
        end
    end
end
verbs = sort(verb);
descriptors = sort([adverb adjective]);
```

PATENT SEARCH QUERY GENERATOR FUNCTION

```
function varargout = patent_search_v2(varargin)
% PATENT_SEARCH_V2 M-file for patent_search_v2.fig
%      PATENT_SEARCH_V2, by itself, creates a new
PATENT_SEARCH_V2 or raises the existing
%      singleton*.
%
%      H = PATENT_SEARCH_V2 returns the handle to a new
PATENT_SEARCH_V2 or the handle to
%      the existing singleton*.
%
%
PATENT_SEARCH_V2('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in PATENT_SEARCH_V2.M with
the given input arguments.
%
%      PATENT_SEARCH_V2('Property','Value',...) creates a
new PATENT_SEARCH_V2 or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before
function_catagorizer_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to
patent_search_v2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
patent_search_v2
```

```
% Last Modified by GUIDE v2.5 18-Nov-2010 21:38:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @patent_search_v2_OpeningFcn, ...
    'gui_OutputFcn',  @patent_search_v2_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before patent_search_v2 is made
visible.
function patent_search_v2_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to patent_search_v2
(see VARARGIN)

% Choose default command line output for patent_search_v2
handles.output = hObject;

handles.query_vector = zeros(1,1700);
handles.index=[];
set(handles.uipanel2,'Visible','off')
temp = load('funct_index.mat');
handles.funct_index = temp.funct_index;
```

```
guidata(hObject, handles);

% UIWAIT makes patent_search_v2 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the
command line.
function varargout = patent_search_v2_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in search.
function search_Callback(hObject, eventdata, handles)
% hObject    handle to search (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
%global out
handles.query_vector = query_vector(1700,handles.index);
handles.output = handles.index;
out = handles.query_vector';
searchviewer_v2(out)
%patent_search_v2_OutputFcn(hObject, eventdata, handles)
%close;

function funct_cat_Callback(h, eventdata, handles, str)
% Get the index_num pointer and the addresses

%
switch str
    case 'branch'
```

```
        %disp('branch')
        set(handles.branch,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Divide')
        set(handles.sf2,'String','Extract')
        set(handles.sf3,'String','Clean')
        set(handles.sf4,'String','Distribute')
        set(handles.sf5,'String','Penetrate')
        set(handles.sf6,'String','Remove')
        set(handles.sf7,'String','Dilute')
        set(handles.sf8,'String','Split')
        set(handles.sf9,'String','Disperse')
        set(handles.sf10,'String','Break')
        set(handles.sf11,'String','Machine')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')

    case 'channel'
        %disp('channel')
        set(handles.channel,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Import')
        set(handles.sf2,'String','Export')
        set(handles.sf3,'String','Transport')
        set(handles.sf4,'String','Transmit')
        set(handles.sf5,'String','Translate')
        set(handles.sf6,'String','Move')
        set(handles.sf7,'String','Rotate')
        set(handles.sf8,'String','Transfer')
        set(handles.sf9,'String','Oscillate')
        set(handles.sf10,'String','Arrange')
        set(handles.sf11,'String','Direct')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
    case 'connect'
        %disp('connect')
        set(handles.connect,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Connect')
```

```
        set(handles.sf2,'String','Couple')
        set(handles.sf3,'String','Mix')
        set(handles.sf4,'String','Mount')
        set(handles.sf5,'String','Apply')
        set(handles.sf6,'String','Add')
        set(handles.sf7,'String','Combine')
        set(handles.sf8,'String','Encounter')
        set(handles.sf9,'String','Bond')

set(handles.sf10,'Visible','off');set(handles.sf10more,'Vis
ible','off')

set(handles.sf11,'Visible','off');set(handles.sf11more,'Vis
ible','off')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
    case 'control'
        %disp('control')
        set(handles.control,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Increase')
        set(handles.sf2,'String','Decrease')
        set(handles.sf3,'String','Increment')
        set(handles.sf4,'String','Decrement')
        set(handles.sf5,'String','Shape')
        set(handles.sf6,'String','Control')
        set(handles.sf7,'String','Form')
        set(handles.sf8,'String','Change')
        set(handles.sf9,'String','Adjust')
        set(handles.sf10,'String','Actuate')

set(handles.sf11,'Visible','off');set(handles.sf11more,'Vis
ible','off')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
    case 'convert'
        %disp('convert')
```

```
        set(handles.convert,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Convert')
        set(handles.sf2,'String','Substitute')
        set(handles.sf3,'String','Transform')
        set(handles.sf4,'String','Produce')
        set(handles.sf5,'String','Condition')
        set(handles.sf6,'String','Treat')
        set(handles.sf7,'String','Modify')

set(handles.sf8,'Visible','off');set(handles.sf8more,'Visib
le','off')

set(handles.sf9,'Visible','off');set(handles.sf9more,'Visib
le','off')

set(handles.sf10,'Visible','off');set(handles.sf10more,'Vis
ible','off')

set(handles.sf11,'Visible','off');set(handles.sf11more,'Vis
ible','off')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
    case 'provision'
        %disp('provide')
        set(handles.provide,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Store')
        set(handles.sf2,'String','Supply')
        set(handles.sf3,'String','Prevent')
        set(handles.sf4,'String','Collect')
        set(handles.sf5,'String','Protect')
        set(handles.sf6,'String','Stop')
        set(handles.sf7,'String','Contain')
        set(handles.sf8,'String','Inhibit')

set(handles.sf9,'Visible','off');set(handles.sf9more,'Visib
le','off')

set(handles.sf10,'Visible','off');set(handles.sf10more,'Vis
ible','off')
```

```
set(handles.sf11,'Visible','off');set(handles.sf11more,'Vis
ible','off')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
    case 'signal'
        %disp('signal')
        set(handles.signal,'BackgroundColor',[0 1 0])
     %   set(handles.sf1,'String','Actuate')
        set(handles.sf1,'String','Indicate')
        set(handles.sf2,'String','Process')
        set(handles.sf3,'String','Display')
        set(handles.sf4,'String','Detect')
        set(handles.sf5,'String','Measure')
        set(handles.sf6,'String','Compare')
        set(handles.sf7,'String','Select')
        set(handles.sf8,'String','Define')
        set(handles.sf9,'String','Determine')
        set(handles.sf10,'String','Monitor')
        set(handles.sf11,'String','Output')
        set(handles.sf12,'String','Calculate')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
    case 'support'
        %disp('support')
        set(handles.support,'BackgroundColor',[0 1 0])
        set(handles.sf1,'String','Place')
        set(handles.sf2,'String','Secure')
        set(handles.sf3,'String','Support')
        set(handles.sf4,'String','Align')
        set(handles.sf5,'String','Anchor')
        set(handles.sf6,'String','Hold')
        set(handles.sf7,'String','Maintain')

set(handles.sf8,'Visible','off');set(handles.sf8more,'Visib
le','off')

set(handles.sf9,'Visible','off');set(handles.sf9more,'Visib
le','off')
```

```
set(handles.sf10,'Visible','off');set(handles.sf10more,'Vis
ible','off')

set(handles.sf11,'Visible','off');set(handles.sf11more,'Vis
ible','off')

set(handles.sf12,'Visible','off');set(handles.sf12more,'Vis
ible','off')

set(handles.sf13,'Visible','off');set(handles.sf13more,'Vis
ible','off')
end

%jjj = query_generator(handles.funct,str);
handles.primary = str;
set(handles.uipanel2,'Visible','on');
guidata(h,handles);

% --- Executes on button press in sf1.
function sf1_Callback(hObject, eventdata, handles)
% hObject    handle to sf1 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf1,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.divide;
    case 'channel'
        index = handles.funct_index.channel.import;
    case 'connect'
        index = handles.funct_index.connect.connect;
    case 'control'
        index = handles.funct_index.control.increase;
    case 'convert'
        index = handles.funct_index.convert.convert;
    case 'provision'
        index = handles.funct_index.provision.store;
    case 'signal'
        index = handles.funct_index.signal.indicate;
```

```
    case 'support'
        index = handles.funct_index.support.place;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf2.
function sf2_Callback(hObject, eventdata, handles)
% hObject    handle to sf2 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf2,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.extract;
    case 'channel'
        index = handles.funct_index.channel.export;
    case 'connect'
        index = handles.funct_index.connect.couple;
    case 'control'
        index = handles.funct_index.control.decrease;
    case 'convert'
        index = handles.funct_index.convert.substitute;
    case 'provision'
        index = handles.funct_index.provision.supply;
    case 'signal'
        index = handles.funct_index.signal.process;
    case 'support'
        index = handles.funct_index.support.secure;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf3.
function sf3_Callback(hObject, eventdata, handles)
% hObject    handle to sf3 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)% ---
```

```
% Executes on button press in sf5.
str = handles.primary;
index_temp = handles.index;
set(handles.sf3,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.clean;
    case 'channel'
        index = handles.funct_index.channel.transport;
    case 'connect'
        index = handles.funct_index.connect.mix;
    case 'control'
        index = handles.funct_index.control.increment;
    case 'convert'
        index = handles.funct_index.convert.transform;
    case 'provision'
        index = handles.funct_index.provision.prevent;
    case 'signal'
        index = handles.funct_index.signal.display;
    case 'support'
        index = handles.funct_index.support.support;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf4.
function sf4_Callback(hObject, eventdata, handles)
% hObject    handle to sf4 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf4,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.distribute;
    case 'channel'
        index = handles.funct_index.channel.transmit;
    case 'connect'
        index = handles.funct_index.connect.mount;
    case 'control'
        index = handles.funct_index.control.decrement;
```

```
    case 'convert'
        index = handles.funct_index.convert.produce;
    case 'provision'
        index = handles.funct_index.provision.collect;
    case 'signal'
        index = handles.funct_index.signal.detect;
    case 'support'
        index = handles.funct_index.support.align;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

function sf5_Callback(hObject, eventdata, handles)
% hObject    handle to sf5 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf5,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.penetrate;
    case 'channel'
        index = handles.funct_index.channel.translate;
    case 'connect'
        index = handles.funct_index.connect.apply;
    case 'control'
        index = handles.funct_index.control.shape;
    case 'convert'
        index = handles.funct_index.convert.condition;
    case 'provision'
        index = handles.funct_index.provision.protect;
    case 'signal'
        index = handles.funct_index.signal.measure;
    case 'support'
        index = handles.funct_index.support.anchor;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf6.
function sf6_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to sf6 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf6,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.remove;
    case 'channel'
        index = handles.funct_index.channel.move;
    case 'connect'
        index = handles.funct_index.connect.add;
    case 'control'
        index = handles.funct_index.control.control;
    case 'convert'
        index = handles.funct_index.convert.treat;
    case 'provision'
        index = handles.funct_index.provision.stop;
    case 'signal'
        index = handles.funct_index.signal.compare;
    case 'support'
        index = handles.funct_index.support.hold;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf7.
function sf7_Callback(hObject, eventdata, handles)
% hObject    handle to sf7 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf7,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.dilute;
    case 'channel'
        index = handles.funct_index.channel.rotate;
```

```
    case 'connect'
        index = handles.funct_index.connect.combine;
    case 'control'
        index = handles.funct_index.control.form;
    case 'convert'
        index = handles.funct_index.convert.modify;
    case 'provision'
        index = handles.funct_index.provision.contain;
    case 'signal'
        index = handles.funct_index.signal.select;
    case 'support'
        index = handles.funct_index.support.maintain;
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf8.
function sf8_Callback(hObject, eventdata, handles)
% hObject    handle to sf8 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf8,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.split;
    case 'channel'
        index = handles.funct_index.channel.transfer;
    case 'connect'
        index = handles.funct_index.connect.encounter;
    case 'control'
        index = handles.funct_index.control.change;
    case 'convert'
        index = [];
    case 'provision'
        index = handles.funct_index.provision.inhibit;
    case 'signal'
        index = handles.funct_index.signal.define;
    case 'support'
        index = [];
end
```

```
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf9.
function sf9_Callback(hObject, eventdata, handles)
% hObject    handle to sf9 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf9,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.disperse;
    case 'channel'
        index = handles.funct_index.channel.oscillate;
    case 'connect'
        index = handles.funct_index.connect.bond;
    case 'control'
        index = handles.funct_index.control.adjust;
    case 'convert'
        index = [];
    case 'provision'
        index = [];
    case 'signal'
        index = handles.funct_index.signal.determine;
    case 'support'
        index = [];
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf10.
function sf10_Callback(hObject, eventdata, handles)
% hObject    handle to sf10 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf10,'BackgroundColor',[0 1 0])
```

```
switch str
    case 'branch'
        index = handles.funct_index.branch.break;
    case 'channel'
        index = handles.funct_index.channel.arrange;
    case 'connect'
        index = [];
    case 'control'
        index =  handles.funct_index.control.actuate;
    case 'convert'
        index = [];
    case 'provision'
        index = [];
    case 'signal'
        index = handles.funct_index.signal.monitor;
    case 'support'
        index = [];
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf11.
function sf11_Callback(hObject, eventdata, handles)
str = handles.primary;
index_temp = handles.index;
set(handles.sf11,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = handles.funct_index.branch.machine;
    case 'channel'
        index = handles.funct_index.channel.direct;
    case 'connect'
        index = [];
    case 'control'
        index = [];
    case 'convert'
        index = [];
    case 'provision'
        index = [];
    case 'signal'
        index = handles.funct_index.signal.output;
    case 'support'
        index = [];
end
```

```
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf12.
function sf12_Callback(hObject, eventdata, handles)
% hObject    handle to sf12 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf12,'BackgroundColor',[0 1 0])
switch str
    case 'branch'
        index = [];
    case 'channel'
        index = [];
    case 'connect'
        index = [];
    case 'control'
        index = [];
    case 'convert'
        index = [];
    case 'provision'
        index = [];
    case 'signal'
        index = handles.funct_index.signal.calculate;
    case 'support'
        index = [];
end
handles.index = [index_temp,index];
guidata(hObject, handles);

% --- Executes on button press in sf13.
function sf13_Callback(hObject, eventdata, handles)
% hObject    handle to sf13 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
str = handles.primary;
index_temp = handles.index;
set(handles.sf13,'BackgroundColor',[0 1 0])
```

```
switch str
    case 'branch'
        index = [];
    case 'channel'
        index = [];
    case 'connect'
        index = [];
    case 'control'
        index = [];
    case 'convert'
        index = [];
    case 'provision'
        index = [];
    case 'signal'
        index = [];
    case 'support'
        index = [];
end
handles.index = [index_temp,index];
guidata(hObject, handles);


% --- Executes on button press in sfdone.
function sfdone_Callback(hObject, eventdata, handles)
% hObject    handle to sfdone (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.uipanel2,'Visible','off')
set(handles.sf1,'Visible','on');
set(handles.sf1,'BackgroundColor',[1 1
1]);set(handles.sf1more,'Visible','on')
set(handles.sf2,'Visible','on');
set(handles.sf2,'BackgroundColor',[1 1 1]);
set(handles.sf2more,'Visible','on')
set(handles.sf3,'Visible','on');
set(handles.sf3,'BackgroundColor',[1 1 1]);
set(handles.sf3more,'Visible','on')
set(handles.sf4,'Visible','on');
set(handles.sf4,'BackgroundColor',[1 1 1]);
set(handles.sf4more,'Visible','on')
```

```
set(handles.sf5,'Visible','on');
set(handles.sf5,'BackgroundColor',[1 1 1]);
set(handles.sf5more,'Visible','on')
set(handles.sf6,'Visible','on');
set(handles.sf6,'BackgroundColor',[1 1 1]);
set(handles.sf6more,'Visible','on')
set(handles.sf7,'Visible','on');
set(handles.sf7,'BackgroundColor',[1 1 1]);
set(handles.sf7more,'Visible','on')
set(handles.sf8,'Visible','on');
set(handles.sf8,'BackgroundColor',[1 1 1]);
set(handles.sf8more,'Visible','on')
set(handles.sf9,'Visible','on');
set(handles.sf9,'BackgroundColor',[1 1 1]);
set(handles.sf9more,'Visible','on')
set(handles.sf10,'Visible','on');
set(handles.sf10,'BackgroundColor',[1 1 1]);
set(handles.sf10more,'Visible','on')
set(handles.sf11,'Visible','on');
set(handles.sf11,'BackgroundColor',[1 1 1]);
set(handles.sf11more,'Visible','on')
set(handles.sf12,'Visible','on');
set(handles.sf12,'BackgroundColor',[1 1 1]);
set(handles.sf12more,'Visible','on')
set(handles.sf13,'Visible','on');
set(handles.sf13,'BackgroundColor',[1 1 1]);
set(handles.sf13more,'Visible','on')


% --- Executes on button press in sf1more.
function more_defs_Callback(hObject, eventdata,
handles,str)
% hObject    handle to sf1more (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
prime = handles.primary;
str;
%disp(char(prime))
switch prime
    case 'branch'
        switch str
            case 'sf1more'
```

```
                index = handles.funct_index.branch.divide;
                secondary = 'Divide';
            case 'sf2more'
                index = handles.funct_index.branch.extract;
                secondary = 'Extract';
            case 'sf3more'
                index = handles.funct_index.branch.clean;
                secondary = 'Clean';
            case 'sf4more'
                index =
handles.funct_index.branch.distribute;
                secondary = 'Distribute';
            case 'sf5more'
                index =
handles.funct_index.branch.penetrate;
                secondary = 'Penetrate';
            case 'sf6more'
                index = handles.funct_index.branch.remove;
                secondary = 'Remove';
            case 'sf7more'
                index = handles.funct_index.branch.dilute;
                secondary = 'Dilute';
            case 'sf8more'
                index = handles.funct_index.branch.split;
                secondary = 'Split';
            case 'sf9more'
                index =
handles.funct_index.branch.disperse;
                secondary = 'Disperse';
            case 'sf10more'
                index = handles.funct_index.branch.break;
                secondary = 'Break';
            case 'sf11more'
                index = handles.funct_index.branch.machine;
                secondary = 'Machine';
        end
    case 'channel'
        switch str
            case 'sf1more'
                index = handles.funct_index.channel.import;
                secondary = 'Import';
            case 'sf2more'
                index = handles.funct_index.channel.export;
                secondary = 'Export';
```

```
            case 'sf3more'
                index =
handles.funct_index.channel.transport;
                secondary = 'Transport';
            case 'sf4more'
                index =
handles.funct_index.channel.transmit;
                secondary = 'Transmit';
            case 'sf5more'
                index =
handles.funct_index.channel.translate;
                secondary = 'Translate';
            case 'sf6more'
                index = handles.funct_index.channel.move;
                secondary = 'Move';
            case 'sf7more'
                index = handles.funct_index.channel.rotate;
                secondary = 'Rotate';
            case 'sf8more'
                index =
handles.funct_index.channel.transfer;
                secondary = 'Transfer';
            case 'sf9more'
                index =
handles.funct_index.channel.oscillate;
                secondary = 'Oscillate';
            case 'sf10more'
                index =
handles.funct_index.channel.arrange;
                secondary = 'Arrange';
            case 'sf11more'
                index = handles.funct_index.channel.direct;
                secondary = 'Direct';
        end
    case 'connect'
        switch str
            case 'sf1more'
                index =
handles.funct_index.connect.connect;
                secondary = 'Connect';
            case 'sf2more'
                index = handles.funct_index.connect.couple;
                secondary = 'Couple';
            case 'sf3more'
```

```
                index = handles.funct_index.connect.mix;
                secondary = 'Mix';
            case 'sf4more'
                index = handles.funct_index.connect.mount;
                secondary = 'Mount';
            case 'sf5more'
                index = handles.funct_index.connect.apply;
                secondary = 'Apply';
            case 'sf6more'
                index = handles.funct_index.connect.add;
                secondary = 'Add';
            case 'sf7more'
                index =
handles.funct_index.connect.combine;
                secondary = 'Combine';
            case 'sf8more'
                index =
handles.funct_index.connect.encounter;
                secondary = 'Encounter';
            case 'sf9more'
                index = handles.funct_index.connect.bond;
                secondary = 'Bond';
        end
    case 'control'
        switch str
            case 'sf1more'
                index =
handles.funct_index.control.increase;
                secondary = 'Increase';
            case 'sf2more'
                index =
handles.funct_index.control.decrease;
                secondary = 'Decrease';
            case 'sf3more'
                index =
handles.funct_index.control.increment;
                secondary = 'Increment';
            case 'sf4more'
                index =
handles.funct_index.control.decrement;
                secondary = 'Decrement';
            case 'sf5more'
                index = handles.funct_index.control.shape;
                secondary = 'Shape';
```

```
            case 'sf6more'
                index =
handles.funct_index.control.control;
                secondary = 'Control';
            case 'sf7more'
                index = handles.funct_index.control.form;
                secondary = 'Form';
            case 'sf8more'
                index = handles.funct_index.control.change;
                secondary = 'Change';
            case 'sf9more'
                index = handles.funct_index.control.adjust;
                secondary = 'Adjust';
            case 'sf10more'
                index =
handles.funct_index.control.actuate;
                secondary = 'Actuate';
        end
    case 'convert'
        switch str
            case 'sf1more'
                index =
handles.funct_index.convert.convert;
                secondary = 'Convert';
            case 'sf2more'
                index =
handles.funct_index.convert.substitute;
                secondary = 'Substitute';
            case 'sf3more'
                index =
handles.funct_index.convert.transform;
                secondary = 'Transform';
            case 'sf4more'
                index =
handles.funct_index.convert.produce;
                secondary = 'Produce';
            case 'sf5more'
                index =
handles.funct_index.convert.condition;
                secondary = 'Condition';
            case 'sf6more'
                index = handles.funct_index.convert.treat;
                secondary = 'Treat';
            case 'sf7more'
```

```
                index = handles.funct_index.convert.modify;
                secondary = 'Modify';
        end
    case 'provision'
        switch str
            case 'sf1more'
                index =
handles.funct_index.provision.store;
                secondary = 'Store';
            case 'sf2more'
                index =
handles.funct_index.provision.supply;
                secondary = 'Supply';
            case 'sf3more'
                index =
handles.funct_index.provision.prevent;
                secondary = 'Prevent';
            case 'sf4more'
                index =
handles.funct_index.provision.collect;
                secondary = 'Collect';
            case 'sf5more'
                index =
handles.funct_index.provision.protect;
                secondary = 'Protect';
            case 'sf6more'
                index = handles.funct_index.provision.stop;
                secondary = 'Stop';
            case 'sf7more'
                index =
handles.funct_index.provision.contain;
                secondary = 'Contain';
            case 'sf8more'
                index =
handles.funct_index.provision.inhibit;
                secondary = 'Inhibit';
        end
    case 'signal'
        switch str
            case 'sf1more'
                index =
handles.funct_index.signal.indicate;
                secondary = 'Indicate';
            case 'sf2more'
```

```
                index = handles.funct_index.signal.process;
                secondary = 'Process';
            case 'sf3more'
                index = handles.funct_index.signal.display;
                secondary = 'Display';
            case 'sf4more'
                index = handles.funct_index.signal.detect;
                secondary = 'Detect';
            case 'sf5more'
                index = handles.funct_index.signal.measure;
                secondary = 'Measure';
            case 'sf6more'
                index = handles.funct_index.signal.compare;
                secondary = 'Compare';
            case 'sf7more'
                index = handles.funct_index.signal.select;
                secondary = 'Select';
            case 'sf8more'
                index = handles.funct_index.signal.define;
                secondary = 'Define';
            case 'sf9more'
                index =
handles.funct_index.signal.determine;
                secondary = 'Determine';
            case 'sf10more'
                index = handles.funct_index.signal.monitor;
                secondary = 'Monitor';
            case 'sf11more'
                index = handles.funct_index.signal.output;
                secondary = 'Output';
            case 'sf12more'
                index =
handles.funct_index.signal.calculate;
                secondary = 'Calculate';
        end
    case 'support'
        switch str
            case 'sf1more'
                index = handles.funct_index.support.place;
                secondary = 'Place';
            case 'sf2more'
                index = handles.funct_index.support.secure;
                secondary = 'Secure';
            case 'sf3more'
```

```
                index =
handles.funct_index.support.support;
                secondary = 'Support';
            case 'sf4more'
                index = handles.funct_index.support.align;
                secondary = 'Align';
            case 'sf5more'
                index = handles.funct_index.support.anchor;
                secondary = 'Anchor';
            case 'sf6more'
                index = handles.funct_index.support.hold;
                secondary = 'Hold';
            case 'sf7more'
                index =
handles.funct_index.support.maintain;
                secondary = 'Maintain';
        end
end
more_display(secondary,sort(index))



% --- Executes on button press in clear.
function clear_Callback(hObject, eventdata, handles)
% hObject    handle to clear (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.branch,'BackgroundColor',[1 1 1])
set(handles.channel,'BackgroundColor',[1 1 1])
set(handles.connect,'BackgroundColor',[1 1 1])
set(handles.control,'BackgroundColor',[1 1 1])
set(handles.convert,'BackgroundColor',[1 1 1])
set(handles.provide,'BackgroundColor',[1 1 1])
set(handles.signal,'BackgroundColor',[1 1 1])
set(handles.support,'BackgroundColor',[1 1 1])
handles.query_vector = zeros(1,1700);
handles.index=[];
guidata(hObject, handles);
```

PATENT SEARCH RESULT VIEWER FUNCTION

```
 function varargout = searchviewer_v2(varargin)
% SEARCHVIEWER_V2 M-file for searchviewer_v2.fig
%      SEARCHVIEWER_V2, by itself, creates a new
SEARCHVIEWER_V2 or raises the existing
%      singleton*.
%
%      H = SEARCHVIEWER_V2 returns the handle to a new
SEARCHVIEWER_V2 or the handle to
%      the existing singleton*.
%
%
SEARCHVIEWER_V2('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in SEARCHVIEWER_V2.M with
the given input arguments.
%
%      SEARCHVIEWER_V2('Property','Value',...) creates a
new SEARCHVIEWER_V2 or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before searchviewer_v2_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to
searchviewer_v2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
searchviewer_v2

% Last Modified by GUIDE v2.5 03-Jan-2011 23:10:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn',
@searchviewer_v2_OpeningFcn, ...
                   'gui_OutputFcn',
@searchviewer_v2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before searchviewer_v2 is made visible.
function searchviewer_v2_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to searchviewer_v2 (see
VARARGIN)

handles.query_vec = varargin{1};

handles.cos_relevancy =
cosine_relevancy(handles.query_vec);
temp = load('fcm_valid.mat');
handles.fcm = temp.fcm';
temp=[];

temp = load('patent_title_valid.mat');
handles.patent_title = temp.patent_title;
temp=[];

temp = load('patent_number_valid.mat');
```

```
handles.patent_number = temp.patent_number;
temp=[];

temp = load('class_valid.mat');
handles.patent_class = temp.class;
temp=[];

temp = load('uspto_classes.mat');
handles.uspto_classes = temp.uspto_classes;
temp=[];
% Choose default command line output for searchviewer_v2

% Update handles structure
handles.output = hObject;
guidata(hObject, handles);
calc_relevancy(hObject, handles);

% UIWAIT makes searchviewer_v2 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the
command line.
function varargout = searchviewer_v2_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in resultbox.
function resultbox_Callback(hObject, eventdata, handles)
% hObject    handle to resultbox (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
```

```
% handles    structure with handles and user data (see
GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
resultbox contents as cell array
%        contents{get(hObject,'Value')} returns selected
item from resultbox
pat_num =
handles.group_number(get(handles.resultbox,'Value'));
tot_rel =
handles.group_tot_rel(get(handles.resultbox,'Value'));
set(handles.tot_rel,'String',num2str(tot_rel))
srch_url =
strcat('http://www.freepatentsonline.com/',num2str(pat_num)
,'.pdf');
web(char(srch_url),'-browser');

% --- Executes during object creation, after setting all
properties.
function resultbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to resultbox (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all
properties.
function alpha_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alpha (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

```
% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all
properties.
function beta_CreateFcn(hObject, eventdata, handles)
% hObject    handle to beta (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all
properties.
function top_xx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to top_xx (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in recompute.
function recompute_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to recompute (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

guidata(hObject, handles);
calc_relevancy(hObject, handles);

% --- Executes on button press in save.
function save_Callback(hObject, eventdata, handles)
% hObject    handle to save (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
SAVE 'test_save.mat' -STRUCT handles result_title
result_number result_class query_vec

% --- Executes on button press.
function change_class_Callback(hObject, eventdata, handles,
str)
% hObject    handle to prev_class (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
switch str
    case 'prev'
        handles.class_track = handles.class_track - 1;
    case 'next'
        handles.class_track = handles.class_track + 1;
end
handles = group_classes(handles);
set(handles.resultbox,'Value',1)
set(handles.resultbox,'String',upper(handles.group_title))
set(handles.class_title,'String',upper(handles.group_class)
)
set(handles.class_disp,'String',num2str(handles.class_track
))
set(handles.class_max,'String',num2str(handles.class_max_nu
m))
guidata(hObject, handles);
```

```
function calc_relevancy(h,handles)

handles.class_track = 1;
handles.a1 = str2num(get(handles.alpha,'String'));
handles.b1 = str2num(get(handles.beta,'String'));
handles.num_result = str2num(get(handles.top_xx,'String'));
x_temp = (1:handles.num_result);
handles.total_relevancy = handles.a1*handles.cos_relevancy
+ handles.b1*handles.fcm;
[rank,index] = sort(handles.total_relevancy,'descend');
handles.indexes = index;
top_fcm = extract(handles.fcm,handles.indexes(x_temp));
cos_rel =
extract(handles.cos_relevancy,handles.indexes(x_temp));
handles.result_title =
extract(handles.patent_title,handles.indexes(x_temp));
handles.result_number =
extract(handles.patent_number,handles.indexes(x_temp));
handles.result_class =
extract(handles.patent_class,handles.indexes(x_temp));
handles.result_cos_rel =
extract(handles.cos_relevancy,handles.indexes(x_temp));
handles.result_tot_rel =
extract(handles.total_relevancy,handles.indexes(x_temp));

handles = group_classes(handles);
top_tot_rel = rank(x_temp);
plot(x_temp,cos_rel(x_temp),x_temp,top_fcm(x_temp),x_temp,t
op_tot_rel(x_temp)),legend('Cosine','FCM','Total')
set(handles.resultbox,'Value',1)
set(handles.resultbox,'String',upper(handles.group_title))
set(handles.class_title,'String',upper(handles.group_class)
)
set(handles.class_disp,'String',num2str(handles.class_track
))
set(handles.class_max,'String',num2str(handles.class_max_nu
m))
guidata(h, handles);
```

Appendix C: TreeTagger Tag Set (Schmid, 1994)

POS Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	/there/ is
FW	foreign word	d'hoevre
IN	preposition, subordinating conjunction	in, of, like
IN/that	/that/ as subordinator	that
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NP	proper noun, singular	John
NPS	proper noun, plural	Vikings
ORD	ordinal number	1
PDT	predeterminer	/both/ the boys
POS	possessive ending	friend/'s/
PP	personal pronoun	I, he, it
PP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give /up /
SENT	Sentence-break punctuation	. ! ?
SYM	Symbol	/ [= *
TO	infinitive 'to'	/to/ go
UH	interjection	uhhuhhuhh
VB	verb /be/, base form	be
VBD	verb /be/, past tense	was, were
VBG	verb /be/, gerund/present participle	being
VBN	verb /be/, past participle	been
VBP	verb /be/, sing. present, non-3d	am, are
VBZ	verb /be/, 3rd person sing. present	is

VH	verb /have/, base form	have
VHD	verb /have/, past tense	had
VHG	verb /have/, gerund/present participle	having
VHN	verb /have/, past participle	had
VHP	verb /have/, sing. present, non-3d	have
VHZ	verb /have/, 3rd person sing. present	has
VV	verb, base form	take
VVD	verb, past tense	took
VVG	verb, gerund/present participle	taking
VVN	verb, past participle	taken
VVP	verb, sing. present, non-3d	take
VVZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when
#	#	#
$	$	$
Quotation marks '		
``	Opening quotation marks	' "
(	Opening brackets	({
)	Closing brackets	) }
,	Comma	,
:	Punctuation	- ; : -- ...

Appendix D: Function Vocabulary

PRIMARY FUNCTION: CHANNEL

Primary	Secondary	Correspondents
Channel	Import	Permit
		Insert
		Input
		Introduce
		Inlet
		Accept
		Admit
		Fetch
		Inflow
		Breathe
		Aspirate
		Import
		Invite
		Ingest
		Invade
		Inhale
		Include
		Obtain
		Receive
		Enter
		Cannulate
		Induct
		Internalize
		Imbibe

Primary	Secondary	Correspondents
Channel	Transmit	Transmit
		Conduct
		Beam
		Impart
		Impose
		Stroke
		Paddle
		Semiconduct
		Overrun
		Tense

Primary	Secondary	Correspondents
Channel	Export	Open
		Eliminate
		Clear
		Omit
		Outlet
		Exit
		Drain
		Waste
		Exhaust
		Flush
		Escape
		Leak
		Empty
		Destroy
		Extrude
		Discard
		Emerge
		Withdraw
		Eject
		Purge
		Evacuate
		Bleed
		Expel
		Dump
		Drip
		Spill
		Expire
		Scrap
		Leach
		Egress
		Export
		Bomb
		Sweat
		Intrude
		Rid
		Trickle
		Excrete
		Exude
		Exhale
		Relinquish
		Vacate
		Destruct
		Ooze
		Obliterate

Primary	Secondary	Correspondents
Channel	Transport	Flow
		Carry
		Port
		Pump
		Transport
		Bring
		Pipe
		Flux
		Pour
		Drift
		Suck
		Transit
		Dig
		Haul
		Glide
		Scoop
		Sling
		Shovel
		Spoon
		Ladle
		Drive
		Tote

Primary	Secondary	Correspondents
Channel	Translate	Drop
		Raise
		Slide
		Shift
		Convey
		Slip
		Translate
		Retract
		Traverse
		Jack
		Migrate
		Telescope
		Ratchet
		Jar
		Skid
		Gusset

Primary	Secondary	Correspondents
Channel	Move	Press
		Advance
		Pull
		Push
		Lift
		Elevate
		Ram
		Lever
		Thrust
		Drag
		Propel
		Trip
		Throw
		Tow
		Cantilever
		Shuttle
		Hoist
		Climb
		Kick
		Plunge
		Move
		Jerk
		Mobilize
		Loft
		Flick
		Tug
		Toss
		Heave

Primary	Secondary	Correspondents
Channel	Transfer	Pass
		Free
		Transfer
		Handle
		Blow
		Relay
		Propagate
		Pocket
		Confer
		Bail
		Vault
		Siphon
		Sip

Primary	Secondary	Correspondents
Channel	Rotate	Rotate
		Angle
		Screw
		Roll
		Coil
		Pivot
		Incline
		Wind
		Belt
		Slope
		Hinge
		Torque
		Wrap
		Pitch
		Tilt
		Spin
		Swing
		Twist
		String
		Flip
		Spiral
		Centrifuge
		Crank
		Reel
		Slant
		Spool
		Articulate
		Swivel
		Revolve
		Wrench
		Curl
		Swirl
		Orientate
		Fringe
		Cock
		Knot
		Tumble
		Orbit
		Winch
		Yaw
		Convolute
		Recline
		Turn
		Whirl
		Freewheel
		Convolve

Primary	Secondary	Correspondents
Channel	Oscillate	Spring
		Circulate
		Reciprocate
		Oscillate
		Rock
		Vibrate
		Flap
		Shake
		Bounce
		Ripple
		Resonate
		Pulsate
		Wobble
		Rebound
		Undulate
		Rattle
		Sway
		Flutter
		Fibrillate
		Chatter

Primary	Secondary	Correspondents	Primary	Secondary	Correspondents
Channel	Direct	Lead	Channel	Arrange	Cross
		Direct			Arrange
		Switch			Oppose
		Guide			Bias
		Channel			Offset
		Curve			Transition
		Stream			Intersect
		Thread			Trail
		Bend			Clutch
		Route			Wrinkle
		Focus			Maneuver
		Deflect			Plumb
		Bypass			Circumvent
		Duct			Linearize
		Steer			Streamline
		Funnel			Diffract
		Emanate			Chase
		Divert			Animate
		Shoot			Kink
		Adduct			Straighten
		Collimate			Twine
		Refract			Tuck
		Pursue			Dangle
		Broaden			Follow
		Defer			Space
		Backflow			Feather
		Dodge			Droop
		Spit			Evert
		Swipe			Rasterize
		Topple			

PRIMARY FUNCTION: BRANCH

Primary	Secondary	Correspondents
Branch	Divide	Section
		Divide
		Segment
		Branch
		Sort
		Partition
		Tab
		Miss
		Diverge
		Fractionate
		Cube
		Segregate
		Dissociate
		Graduate
		Quantize
		Parse
		Allot
		Buck
		Dissect
		Dismantle
		Shred
		Interdigitate
		Packetize
		Compartmentalize
		Part
		Separate
		Digitalize
		Modularize
		Butcher
		Sectionalize

Primary	Secondary	Correspondents
Branch	Penetrate	Penetrate
		Perforate
		Punch
		Pierce
		Shank
		Spike
		Pit
		Puncture
		Lance
		Spear
		Gore
		Stab
		Stipple
		Crater

Primary	Secondary	Correspondents
Branch	Extract	Filter
		Screen
		Mask
		Suction
		Filtrate
		Differentiate
		Reject
		Purify
		Sieve
		Comb
		Degas
		Weed
		Rake
		Sift
		Outgas
		Dab
		Eradicate
		Cull
		Desiccate

Primary	Secondary	Correspondents
Branch	Distribute	Distribute
		Network
		Spread
		Share
		Fork
		Shunt
		Plow
		Disseminate
		Shuffle
		Reticulate
		Portion

Primary	Secondary	Correspondents
Branch	Dilute	Dilute
		Diffuse
		Dissipate
		Refine
		Mist
		Froth
		Smudge
		Equispace
		Ravel

Primary	Secondary	Correspondents
Branch	Clean	Strip
		Vacuum
		Clean
		Wash
		Dust
		Polish
		Rinse
		Brush
		Elute
		Sweep
		Sterilize
		Wipe
		Scrape
		Bleach
		Scrub
		Abrade
		Cleanse
		Bathe
		Buff
		Mop
		Swab
		Squeegee
		Scour
		Sanitize
		Exfoliate
		Floss
		Chafe

Primary	Secondary	Correspondents
Branch	Remove	Core
		Wear
		Etch
		Skin
		Subtract
		Trim
		Peel
		Scratch
		Rub
		Tunnel
		Crop
		Trench
		Excise
		Shed
		Shave
		Erode
		Skim
		Engrave
		Ablate
		Excavate
		Abolish
		Prune
		Scuff
		Fret
		Spall
		Graze
		Photoetch
		Husk
		Slough
		Itch

Primary	Secondary	Correspondents
Branch	Disperse	Explode
		Disperse
		Spray
		Fan
		Cascade
		Rain
		Scatter
		Shower
		Splash
		Sprinkle
		Bristle
		Apportion
		Fray
		Singulate
		Splay
		Disburse
		Nebulize

Primary	Secondary	Correspondents
Branch	Machine	Face
		Machine
		Cut
		Plane
		Tool
		Lap
		Bore
		Mill
		Drill
		Tap
		Grind
		Sand
		Saw
		Lathe
		Ream
		Rifle
		Surface
		Sandblast
		Hone
		Chisel
		Broach
		Miter
		Mortise
		Deburr

Primary	Secondary	Correspondents
Branch	Break	Break
		Powder
		Chip
		Snap
		Crack
		Fragment
		Tear
		Fracture
		Granulate
		Pulp
		Flake
		Pulverize
		Sever
		Triturate
		Comminute
		Pelletize
		Rip
		Mow
		Crumble
		Shatter
		Mince

Primary	Secondary	Correspondents
Branch	Split	Groove
		Split
		Shear
		Slit
		Detach
		Burst
		Slice
		Seam
		Rupture
		Cleave
		Grate
		Pop
		Chop
		Bifurcate
		Bisect
		Nick
		Blister
		Breach
		Fissure
		Slash
		Carve
		Incise
		Binarize
		Kerf

PRIMARY FUNCTION: CONNECT

Primary	Secondary	Correspondents
Connect	Connect	Contact
		Reach
		Link
		Join
		Bridge
		Bind
		Touch
		Span
		Conjugate
		Impinge
		Beat
		Ligate
		Collide
		Retrofit
		Dock
		Connect
		Conjoin

Primary	Secondary	Correspondents
Connect	Mix	Mix
		Stir
		Blend
		Slurry
		Agitate
		Admix
		Emulsify
		Knit
		Weave
		Tangle
		Braid
		Intersperse
		Scramble
		Mingle
		Sparge
		Perfuse
		Whip
		Percolate

Primary	Secondary	Correspondents
Connect	Couple	Associate
		Couple
		Group
		Approach
		Match
		Mate
		Unite
		Sandwich
		Merge
		Converge
		Attract
		Aggregate
		Cluster
		Shadow
		Agglomerate
		Lump
		Clump
		Bunch
		Bale
		Relate
		Commingle
		Coax
		Compatibilize
		Nestle
		Herd

Primary	Secondary	Correspondents
Connect	Apply	Plate
		Layer
		Apply
		Coat
		Affix
		Laminate
		Deposit
		Till
		Overlay
		Ply
		Clad
		Tile
		Imprint
		Sputter
		Plaster
		Underlay

Primary	Secondary	Correspondents
Connect	Mount	Mount
		Attach
		Wire
		Assemble
		Chain
		Install
		Bolt
		Hook
		Button
		Clip
		Tie
		Adjoin
		Rivet
		Yoke
		Fasten
		Stitch
		Staple
		Nail
		Rope
		Sew
		Lace
		Tack
		Suture
		Hitch
		Zip
		Dovetail
		Equip
		Cling
		Reeve
		Grout
		Dart
		Shackle
		Lash
		Quilt
		Snare
		Piggyback

Primary	Secondary	Correspondents
Connect	Add	Add
		Incorporate
		Append
		Absorb
		Stack
		Pack
		Dope
		Contaminate
		Patch
		Graft
		Lodge
		Cake
		Pile
		Pollute
		Infuse
		Infiltrate
		Insel
		Congest
		Infesl
		Heap
		Instill

Primary	Secondary	Correspondents
Connect	Combine	Combine
		Integrate
		Compose
		Alloy
		Fmbed
		Mesh
		Fuse
		Implant
		Impregnate
		Annex
		Interleave
		Splice
		Compile
		Consolidate
		Concatenate
		Collate
		Flocculate
		Unify
		Assimilate
		Palletize
		Colonize
		Prill

Primary	Secondary	Correspondents
Connect	Encounter	Encounter
		Hammer
		Batter
		Tamp
		Slam
		Impact
		Hit

Primary	Secondary	Correspondents
Connect	Bond	Bond
		Weld
		Adhere
		Paste
		Glue
		Cement
		Solder
		Braze
		Sinter
		Spunbond

PRIMARY FUNCTION: CONVERT

Primary	Secondary	Correspondents
Convert	Convert	React
		Dissolve
		Consume
		Fire
		Polymerize
		Crystallize
		Ionize
		Crosslink
		Cook
		Catalyze
		Digest
		Bake
		Thermoset
		Decay
		Smoke
		Fluidize
		Fume
		Extinguish
		Coagulate
		Transduce
		Mutate
		Volatilize
		Metabolize
		Fry
		Spoil
		Rot
		Ferment
		Fluoresce
		Detonate
		Smelt
		Gelatinize
		Incinerate
		Compost
		Embrittle
		Tarnish

Primary	Secondary	Correspondents
Convert	Produce	Generate
		Manufacture
		Complete
		Water
		Create
		Construct
		Constitute
		Yield
		Fabricate
		Develop
		Synthesize
		Originate
		Seed
		Culture
		Duplicate
		Clone
		Replicate
		Cultivate
		Nucleate
		Proliferate
		Hemorrhage
		Germinate
		Fertilize
		Make
		Produce
		Salvage
		Sow
		Pollinate
		Materialize
		Spawn
		Sprout
		Mulch

Primary	Secondary	Correspondents
Convert	Transform	Reflect
		Melt
		Build
		Mirror
		Gel
		Transform
		Foam
		Evaporate
		Boil
		Oxidize
		Ice
		Steam
		Condense
		Freeze
		Vaporize
		Solubilize
		Spark
		Solidify
		Miniaturize
		Revert
		Metallize
		Atomize
		Frost
		Plasticize
		Vulcanize
		Clot
		Combust
		Dilate
		Acidify
		Liquefy
		Rubberize
		Gasify
		Sublimate
		Disarm
		Congeal

Primary	Secondary	Correspondents
Convert	Condition	Configure
		Deteriorate
		Hydrate
		Starch
		Wax
		Lubricate
		Incubate
		Quench
		Tan
		Cream
		Grease
		Autoclave
		Soak
		Chill
		Brine
		Steep
		Equilibrate
		Sensitize
		Decant
		Magnetize
		Thaw
		Aerate
		Oxygenate
		Radiolabel
		Mineralize
		Pickle
		Odorize
		Condition
		Economize
		Electrolyze
		Pasteurize
		Elasticize

Primary	Secondary	Correspondents
Convert	Substitute	Compound
		Substitute
		Exchange
		Trade
		Sacrifice
		Liberate
		Transplant
		Swap
		Transpose
		Disguise
		Supersede

Primary	Secondary	Correspondents
Convert	Treat	Treat
		Dry
		Service
		Concentrate
		Wet
		Precipitate
		Cast
		Cure
		Dye
		Repair
		Pigment
		Paint
		Nitride
		Irradiate
		Anneal
		Radiate
		Stain
		Infect
		Blast
		Enrich
		Ignite
		Homogenize
		Heal
		Inoculate
		Waterproof
		Electroplate
		Clarify
		Lacquer
		Glaze
		Torch
		Passivate
		Varnish
		Anodize
		Galvanize
		Bombard
		Temper
		Lase
		Colorize
		Humidify
		Moisturize
		Tackify
		Carburize
		Camouflage

Primary	Secondary	Correspondents
Convert	Modify	Heat
		Adapt
		Cool
		Modify
		Invert
		Render
		Pressurize
		Burn
		Flash
		Warm
		Formulate
		Bubble
		Neutralize
		Microwave
		Weather
		Polarize
		Swell
		Refrigerate
		Harden
		Denature
		Strengthen
		Rust
		Soften
		Strobe
		Corrode
		Solvate
		Tint
		Grill
		Coalesce
		Char
		Desorb
		Carbonize
		Roast
		Scorch
		Opacify
		Insolubilize
		Barbecue
		Discolor
		Singe
		Permeabilize
		Toxify
		Siliconize
		Vascularize

PRIMARY FUNCTION: CONTROL

Primary	Secondary	Correspondents
Control	Increase	Facilitate
		Enlarge
		Elongate
		Gain
		Tension
		Accelerate
		Maximize
		Jet
		Amplify
		Augment
		Exaggerate
		Widen
		Expedite
		Magnify
		Intensify
		Stiffen
		Accentuate
		Increase
		Extend
		Densify
		Fluff
		Protract
		Escalate
		Hasten

Primary	Secondary	Correspondents
Control	Increment	Enhance
		Exceed
		Expand
		Precede
		Peak
		Strain
		Grow
		Increment
		Prolong
		Supplement
		Ramp
		Emphasize
		Boost
		Dominate
		Surge
		Lengthen
		Stoke
		Thicken
		Improve
		Heighten
		Overdose

Primary	Secondary	Correspondents
Control	Decrease	Minimize
		Ease
		Restrict
		Dip
		Contract
		Relieve
		Cushion
		Retard
		Shrink
		Relax
		Shorten
		Baffle
		Fatigue
		Deplete
		Fade
		Slack
		Decelerate
		Loosen
		Constrict
		Weaken
		Quiet
		Dim
		Subside
		Lower
		Sap
		Darken

Primary	Secondary	Correspondents
Control	Decrement	Damage
		Compress
		Slow
		Mar
		Suppress
		Lose
		Impair
		Collapse
		Diminish
		Sink
		Attenuate
		Alleviate
		Damp
		Disrupt
		Decrement
		Harm
		Lessen
		Creep
		Dull
		Dampen
		Anesthetize
		Silence
		Recede
		Hamper
		Reduce
		Muffle
		Inflict
		Atrophy
		Lighten

Primary	Secondary	Correspondents
Control	Change	Vary
		Affect
		Scale
		Reverse
		Saturate
		Alter
		Shock
		Localize
		Disturb
		Deviate
		Normalize
		Exploit
		Buckle
		Warp
		Fluctuate
		Ventilate
		Rectify
		Blur
		Irritate
		Aggravate
		Commutate
		Pry
		Line
		Change
		Speckle
		Mottle
		Angulate
		Roughen
		Wilt
		Bruise
		Gravitate

Primary	Secondary	Correspondents
Control	Shape	Crease
		Pattern
		Square
		Narrow
		Smooth
		Round
		Ball
		Iron
		Notch
		Stretch
		Contour
		Taper
		Texture
		Distort
		Pinch
		Stripe
		Bevel
		Skew
		Blunt
		Sag
		Serrate
		Knurl
		Streak
		Dent
		Dimple
		Scar
		Pleat
		Chamfer
		Pave
		Fillet
		Shape
		Distend
		Wad
		Sharpen
		Gouge
		Tamp
		Peen
		Texturize
		Slam
		Crumple
		Sculpt

Primary	Secondary	Correspondents
Control	Control	Enable
		Limit
		Avoid
		Mediate
		Repeat
		Cycle
		Stage
		Sequence
		Alternate
		Attempt
		Delay
		Induce
		Gate
		Suspend
		Regulate
		Execute
		Interfere
		Meter
		Optimize
		Manipulate
		Manage
		Disable
		Sustain
		Moderate
		Throttle
		Lag
		Prod
		Persist
		Pace
		Iterate
		Control
		Evade
		Soothe

Primary	Secondary	Correspondents
Control	Form	Force
		Work
		Mold
		Compact
		Fold
		Tip
		Stress
		Protrude
		Wound
		Conform
		Flex
		Strike
		Rim
		Stamp
		Squeeze
		Crush
		Corrugate
		Hammer
		Emboss
		Lineate
		Bulge
		Knead
		Planarize
		Forge
		Honeycomb
		Smear
		Flatten
		Chew
		Swage
		Stencil
		Thermoform
		Downsize
		Circumflex
		Form
		Furrow
		Electroform
		Squash
		Mash
		Jut
		Burnish
		Beamform

Primary	Secondary	Correspondents
Control	Actuate	Start
		Begin
		Activate
		Actuate
		Initiate
		Try
		Command
		Respond
		Request
		Coordinate
		Synchronize
		Aim
		Trigger
		Prime
		Commence
		Anticipate
		Excite
		Initialize
		Stimulate
		Pilot
		Resume
		Toggle
		Override
		Launch
		Refresh
		Perturb
		Provoke
		Allow
		Operate
		Impel
		Enact

Primary	Secondary	Correspondents
Control	Adjust	Adjust
		Correct
		Influence
		Proportion
		Compensate
		Modulate
		Tune
		Govern
		Equalize
		Tolerate
		Bump
		Mitigate
		Indent
		Leverage
		Hem
		Size
		Postpone
		Stagger
		Jolt
		Encroach
		Harmonize

PRIMARY FUNCTION: PROVISION

Primary	Secondary	Correspondents
Provision	Store	Store
		File
		Accommodate
		Stock
		Preserve
		Bank
		Settle
		Reserve
		Barrel
		Populate
		Archive
		Stuff
		Warehouse
		Stow
		Swallow
		Case
		Captivate
		Sequester

Primary	Secondary	Correspondents
Provision	Collect	Load
		Charge
		Fill
		Sample
		Collect
		Accumulate
		Capture
		Acquire
		Pool
		Nest
		Harvest
		Inflate
		Burden
		Blot
		Gather
		Taste
		Sack
		Scavenge
		Accrue
		Seep
		Procure
		Puddle
		Glean
		Thresh

Primary	Secondary	Correspondents
Provision	Supply	Power
		Appropriate
		Supply
		Release
		Establish
		Feed
		Issue
		Extract
		Provision
		Exert
		Demand
		Inject
		Fuel
		Deliver
		Dispense
		Energize
		Retrieve
		Dedicate
		Allocate
		Deploy
		Counteract
		Replenish
		Secrete
		Permeate
		Administer
		Transfect
		Expend
		Devote
		Ration
		Vend
		Titrate
		Sonicate
		Donate
		Irrigate
		Provide
		Require
		Borrow
		Backfill
		Backpressure
		Drench

Primary	Secondary	Correspondents
Provision	Prevent	Block
		Bar
		Resist
		Trap
		Cancel
		Obstruct
		Obscure
		Intercept
		Decline
		Arrest
		Hide
		Stall
		Repel
		Abort
		Photomask
		Deny
		Starve
		Prevent
		Blockade
		Abate
		Avert
		Eclipse
		Incapacitate

Primary	Secondary	Correspondents
Provision	Inhibit	Inhibit
		Impede
		Inactivate
		Hinder
		Prohibit
		Occlude
		Choke
		Discourage
		Deter
		Deprive
		Curb
		Curtail
		Fireproof
		Veil
		Stagnate

Primary	Secondary	Correspondents	Primary	Secondary	Correspondents	Primary	Secondary	Correspondents
Provision	Stop	Stop	Provision	Contain	Wall	Provision	Protect	Cover
		Terminate			Seal			Protect
		Isolate			Surround			Insulate
		Finish			Chamber			Pad
		Cap			Box			Shield
		Plug			Package			Reinforce
		Interrupt			Circle			Withstand
		Park			Possess			Immerse
		Exclude			Encompass			Capsulate
		Brake			Shell			Shade
		Shut			Confine			Guard
		Obviate			Cup			Sheath
		Preclude			Bag			Conserve
		Cease			Bottle			Jacket
		Gasket			Occupy			Shutter
		Jam			Hatch			Conceal
		Clog			Cage			Flood
		Halt			Hood			Shroud
		Discontinue			Blanket			Immunize
		Neglect			Dam			Slag
		Pause			Crowd			Rescue
		Negate			Envelop			Bandage
		Snag			Bury			Blink
		Mute			Crate			Endure
		End			Close			Shelter
		Quit			Contain			Weatherproof
		Nullify			Submerse			
		Abandon						

PRIMARY FUNCTION: SIGNAL

Primary	Secondary	Correspondents
Signal	Indicate	Signal
		Communicate
		Mark
		Date
		Express
		Denote
		Chart
		Flag
		Instruct
		Tag
		Brand
		Highlight
		Signify
		Announce
		Downlink
		Stipulate
		Serialize
		Number
		Point
		Indicate
		Demarcate

Primary	Secondary	Correspondents
Signal	Detect	Read
		Detect
		Sense
		Image
		Search
		Scan
		Target
		Register
		Probe
		Feel
		Perceive
		Acknowledge
		Tally
		Photocopy
		Inherit

Primary	Secondary	Correspondents
Signal	Process	Program
		Analyze
		Format
		Save
		Automate
		Copy
		Simplify
		Delete
		Organize
		Digitize
		Truncate
		Standardize
		Erase
		Recall
		Skip
		Interpret
		Transcribe
		Encrypt
		Corrupt
		Randomize
		Decrypt
		Invalidate
		Bioassay
		Forecast
		Process
		Decipher
		Penalize

Primary	Secondary	Correspondents
Signal	Display	Draw
		Expose
		Display
		Project
		Depict
		Exhibit
		Reveal
		Plot
		Visualize
		Present
		View
		Show
		Illustrate

Primary	Secondary	Correspondents
Signal	Measure	Value
		Measure
		Gage
		Dimension
		Observe
		Identify
		Profile
		Examine
		Map
		Gauge
		Weigh
		Survey
		Quantify
		Caliper
		Time

Primary	Secondary	Correspondents
Signal	Compare	Compare
		Rate
		Pare
		Contrast
		Model
		Check
		Evaluate
		Approximate
		Estimate
		Simulate
		Calibrate
		Verify
		Assess
		Trend
		Validate
		Sync
		Mimic
		Emulate
		Equate
		Reconcile
		Imitate
		Contradict

Primary	Secondary	Correspondents
Signal	Select	Designate
		Implement
		Assign
		Distinguish
		Pick
		Choose
		Decide
		Favor
		Query
		Queue
		Disregard
		Select

Primary	Secondary	Correspondents
Signal	Define	Characterize
		Code
		Grade
		Index
		Label
		Specify
		Script
		Classify
		Categorize
		Quantitate
		Catalogue
		Qualify
		Inscribe
		Define

Primary	Secondary	Correspondents
Signal	Determine	Test
		Access
		Derive
		Recognize
		Confirm
		Resolve
		Schedule
		Seek
		Judge
		Investigate
		Discriminate
		Authorize
		Diagnose
		Rank
		Infer
		Discern
		Violate
		Determine
		Triangulate

Primary	Secondary	Correspondents
Signal	Monitor	Record
		Buffer
		Track
		Monitor
		Tape
		Letter
		Trace
		Log
		Elapse
		Inspect
		Inventory
		Browse
		Type
		Dub

Primary	Secondary	Correspondents
Signal	Output	Light
		Output
		Print
		Page
		Emit
		Send
		Write
		Illuminate
		Multiplex
		Broadcast
		Submit
		Alert
		Draft
		Backlight
		Replay
		Evoke

Primary	Secondary	Correspondents
Signal	Calculate	Difference
		Factor
		Calculate
		Average
		Count
		Sum
		Compute
		Multiply
		Interpolate
		Tabulate
		Extrapolate
		Downconvert

PRIMARY FUNCTION: SUPPORT

Primary	Secondary	Correspondents
Support	Place	Pose
		Ground
		Situate
		Stick
		Place
		Position
		Set
		Emplace

Primary	Secondary	Correspondents
Support	Secure	Secure
		Lock
		Plant
		Latch
		Catch
		Immobilize
		Crimp
		Tighten
		Snug
		Clasp
		Tackle
		Rigidify
		Padlock
		Cinch

Primary	Secondary	Correspondents
Support	Anchor	Fit
		Seat
		Shoulder
		Hang
		Anchor
		Harness
		Stake
		Spoke
		Drape
		Prop
		Tether
		Moor

Primary	Secondary	Correspondents
Support	Support	Support
		Level
		Frame
		Assist
		Wedge
		Rack
		Fixture
		Brace
		Shim
		Scaffold
		Bolster
		Splint
		Buttress

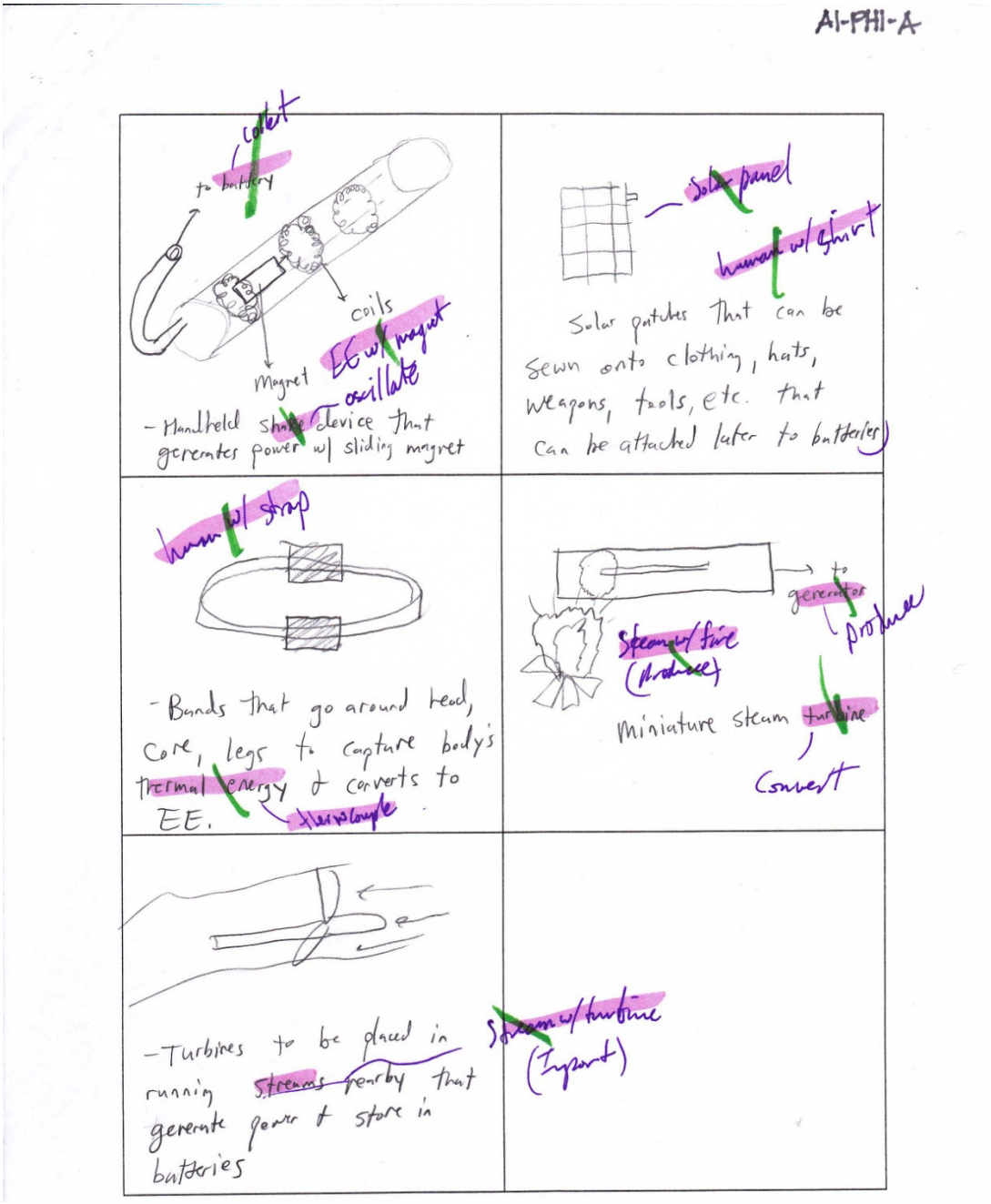
Primary	Secondary	Correspondents
Support	Align	Locate
		Right
		Center
		Align
		Orient
		Upstand
		Centralize
		Posture
		Poise
		Order

Primary	Secondary	Correspondents
Support	Maintain	Maintain
		Rest
		Balance
		Stabilize
		Float
		Steady
		Straddle
		Cradle
		Saddle

Primary	Secondary	Correspondents
Support	Hold	Fix
		Hold
		Retain
		Pin
		Clamp
		Bracket
		Grip
		Grasp
		Strap
		Constrain
		Chuck
		Grab
		Seize
		Clinch
		Grapple

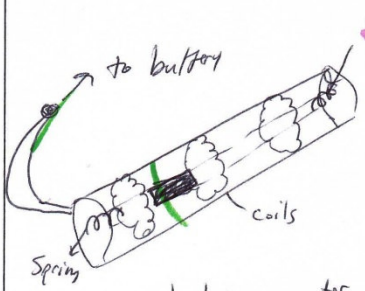
Appendix E: Experimental Results for Patent Analogy Ideation Study

GROUP A



wind w/ windmill

- Miniature wind turbines set up in villages.



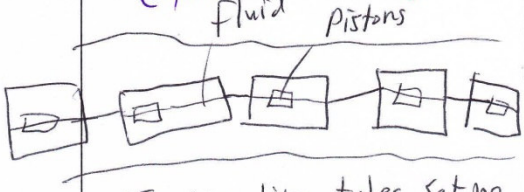
Spring oscillates

EG w/ Induction

- Same shake generator, but w/ springs at each end (inside) to aid in moving the magnet back & forth

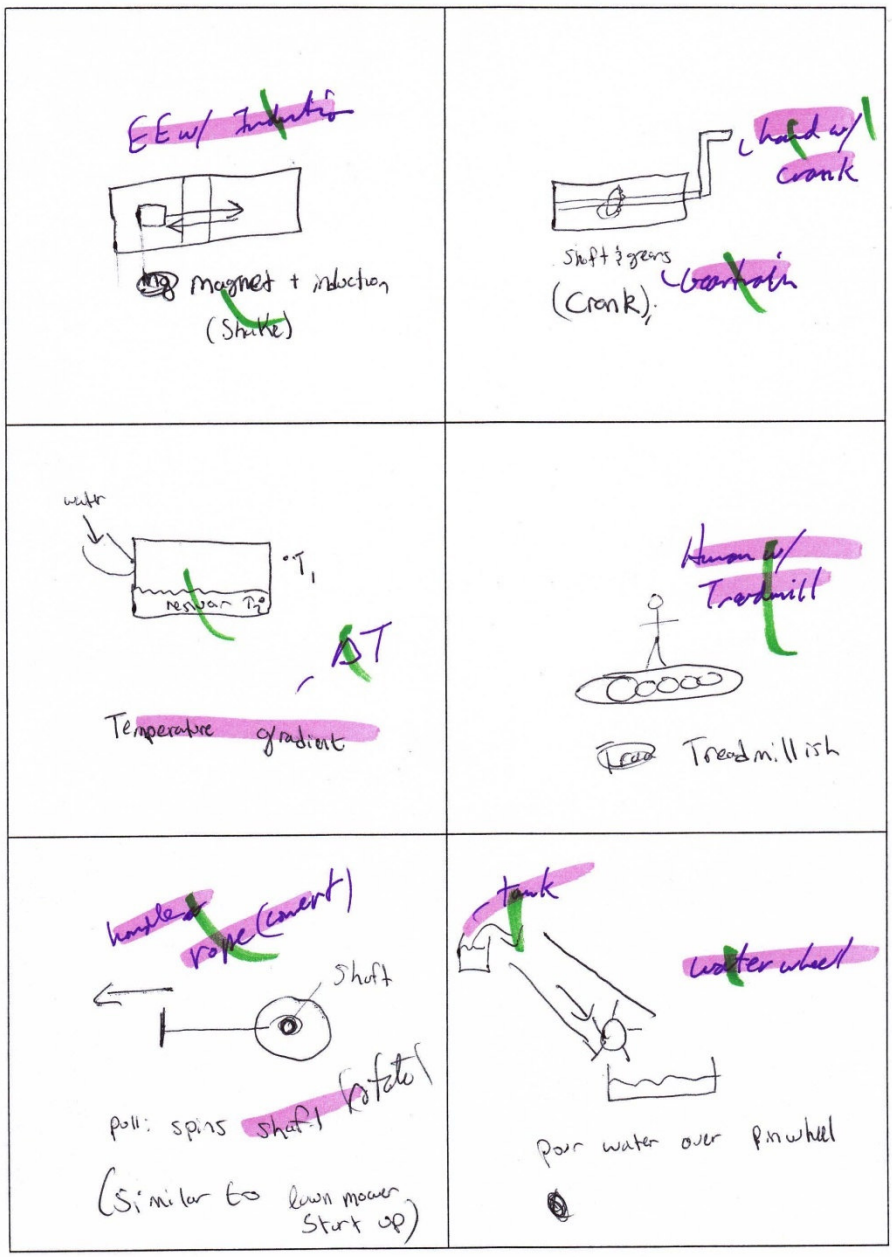
waves/ float (Input)

pressure w/ piston

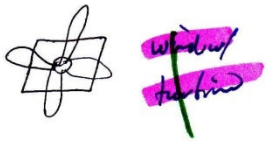


Snake-like tubes set up in rivers. The waves & disturbances cause each cell to fluctuate & therefore each piston to pump oil or fluid in one direction.

A2-PHI-A

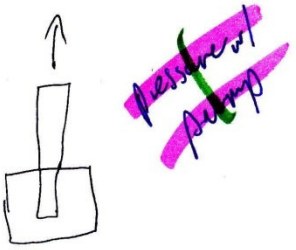

EE w/ Induction
Magnet + induction
(Stroke)
Hand w/ crank
Shaft & gears
(Crank)
Gearbox
water
reservoir T₀
T₁
ΔT
Temperature gradient
Human / Treadmill
Treadmill ish
handle
rope (cord)
Shaft
pull: spins shaft (rotate)
(Similar to lawn mower start up)
tank
water wheel
pour water over pinwheel

A2-PHI-B



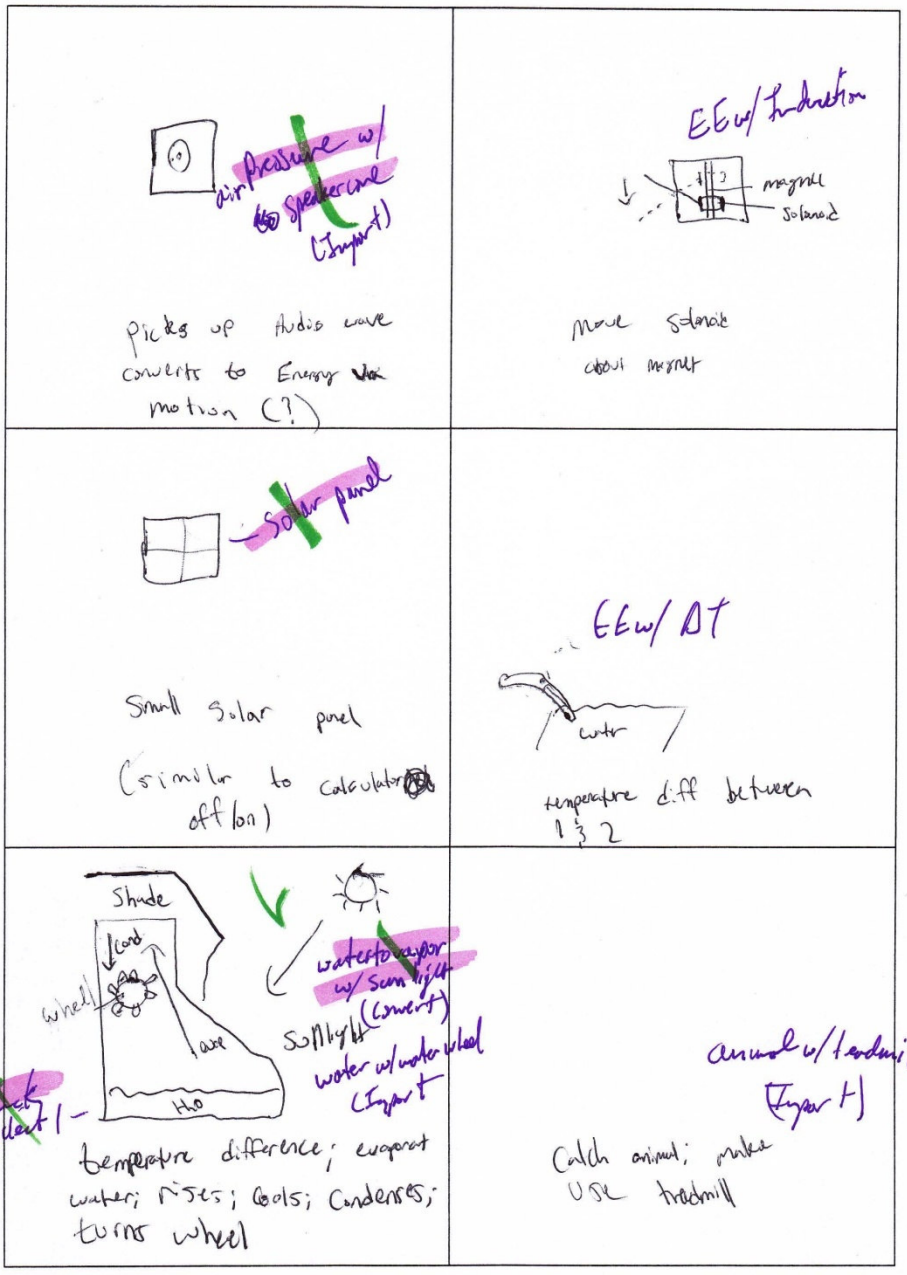
Small wind turbine

backpack w/ water in it, "waves" generated by walking

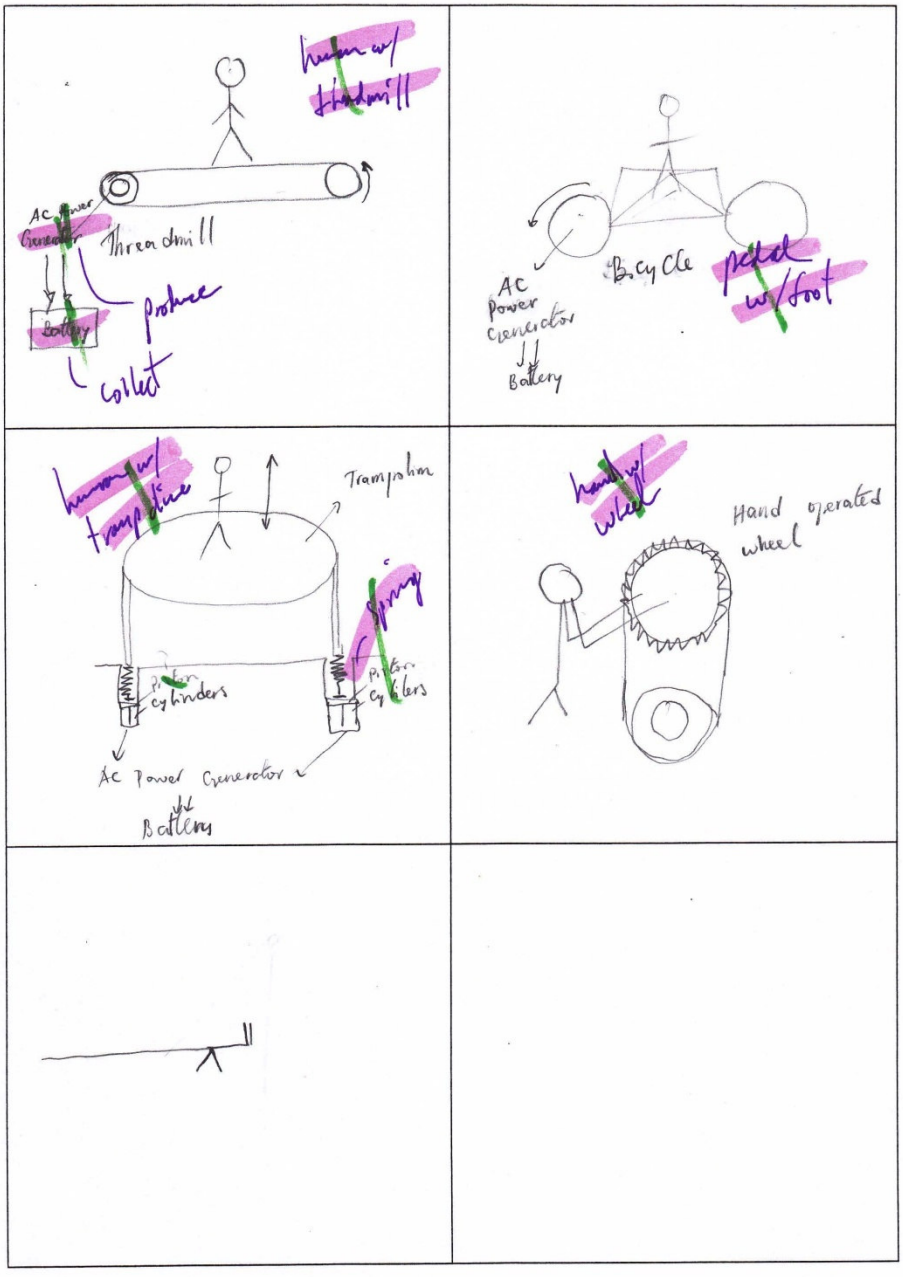


pumping system

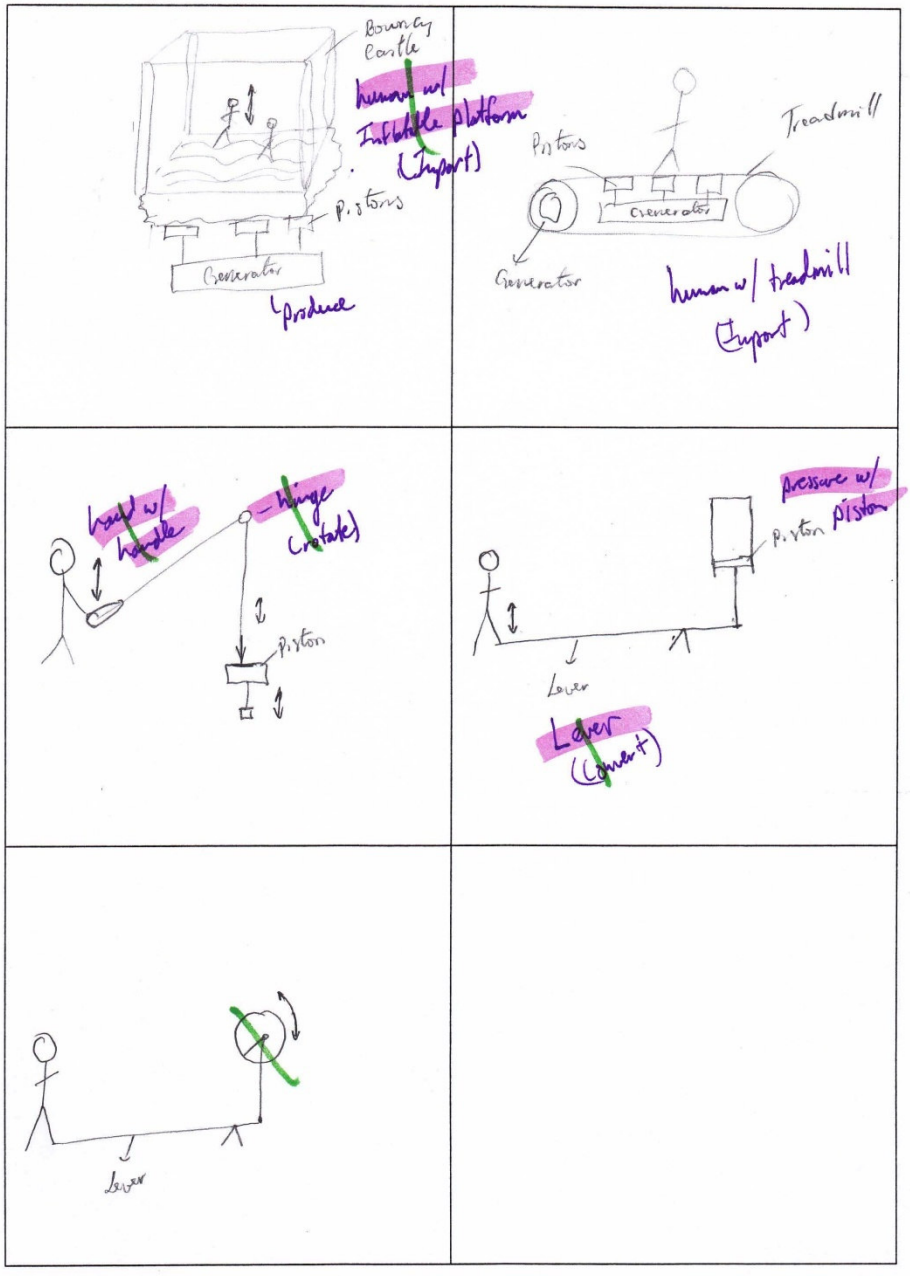
A2-PHB-A

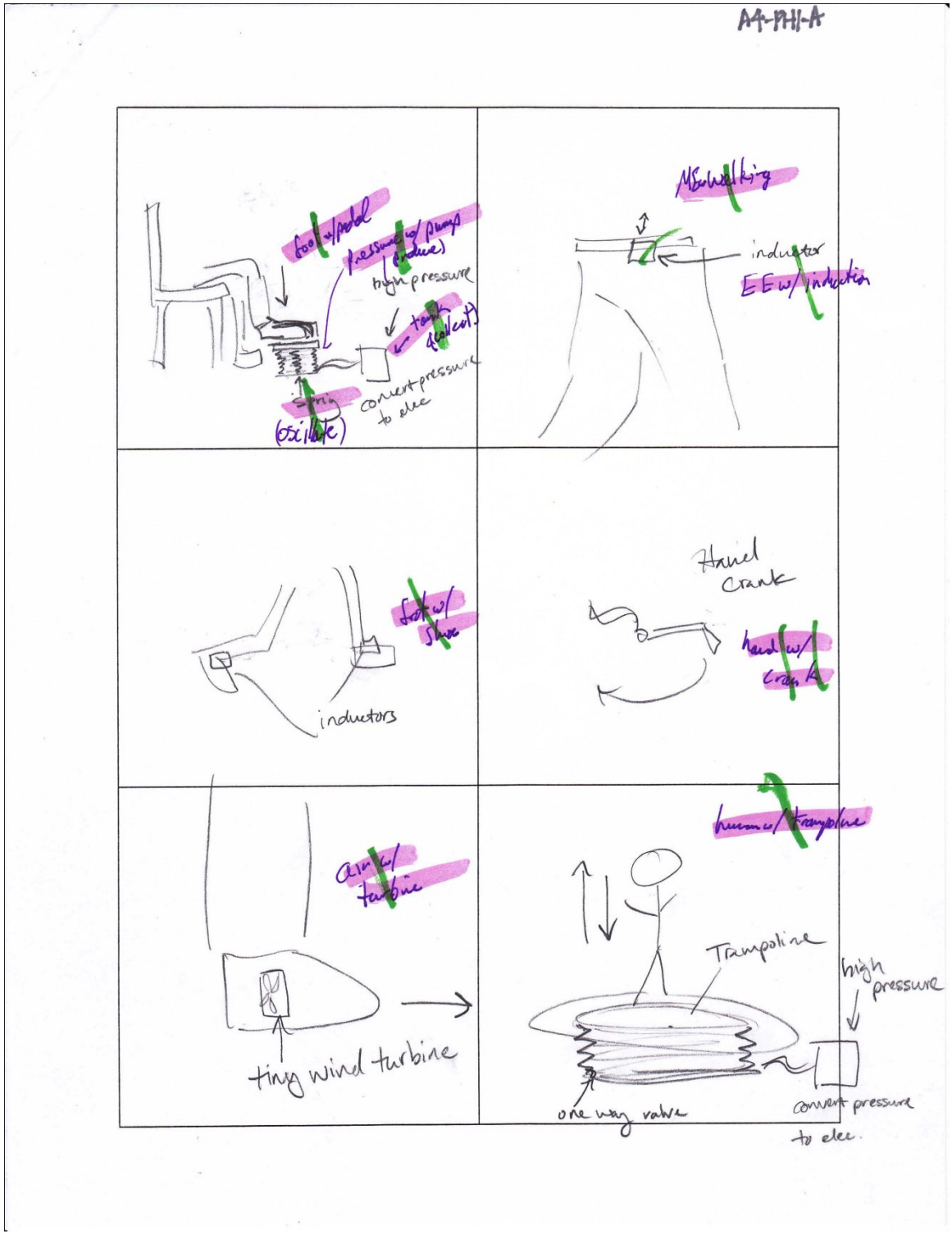

air pressure w/ speaker cone (import)
picks up Audio wave converts to Energy motion (?)
EEw/ Induction
magnet
Solenoid
move Solenoid about magnet
Solar panel
Small Solar panel
(similar to calculator off/on)
EEw/ ΔT
water
temperature diff between 1 & 2
Shade
load
wheel
axle
H2O
water to vapor w/ semi-flexible (convert)
Softlight
water w/ water wheel (import)
Tank Collect
temperature difference; evaporat water; rises; cools; condenses; turns wheel
animal w/ treadmill (import)
Catch animal; make use treadmill

AS-PHI-A

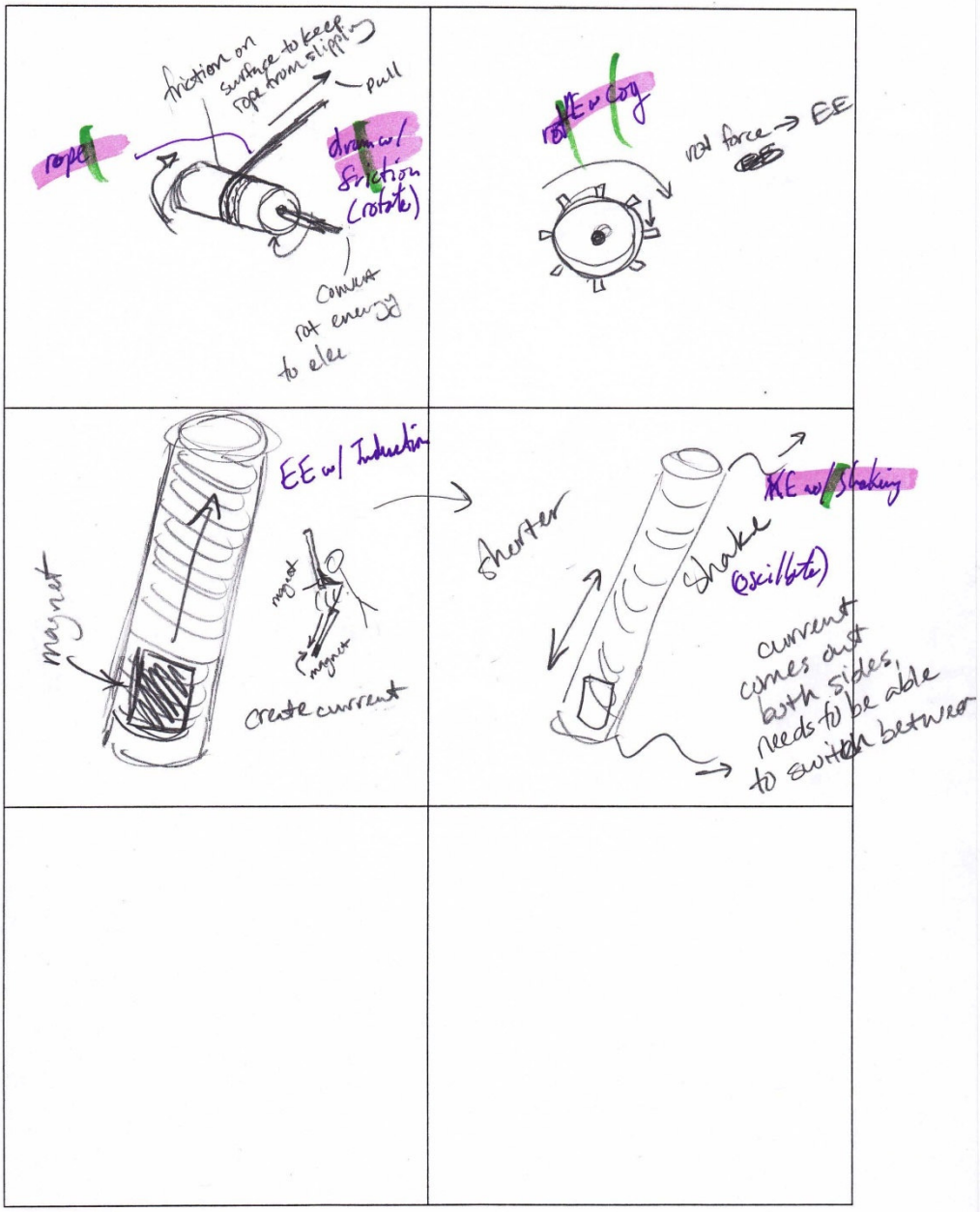

human w/ treadmill
AC Power Generator
Threadmill
Battery
produce
collect
Bicycle
AC Power Generator
Battery
pedal w/ foot
human w/ trampoline
Trampoline
Piston cylinders
Spring
Piston Cylinders
AC Power Generator
Battery
hand w/ wheel
Hand operated wheel

A3-PH3-A


Bouncy Castle
human w/ Inflatable platform (Input)
Pistons
Generator
produce
Treadmill
Pistons
Generator
human w/ treadmill (Input)
hand w/ handle
hinge (rotate)
piston
pressure w/ piston piston
Lever
Lever (Input)
Lever

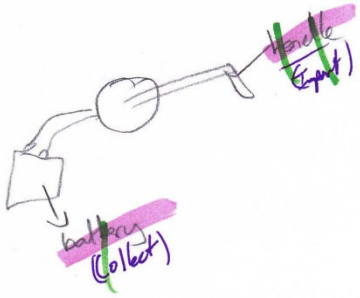

A4-PHI-A
foot pedal
pressure w/ pump
high pressure
convert pressure
to elec
(oscillate)
industry
EE w/ induction
inductors
Hand
Crank
ting wind turbine
Trampoline
high
pressure
one way valve
convert pressure
to elec.

A4-PHS-A


friction on surface to keep rope from slipping
pull
rope
drum w/ friction (rotate)
convert KE energy to elec
KE w/ Coy
net force → EE
EE w/ Induction
magnet
magnet
magnet
create current
shorter
KE w/ Shaking
shake (oscillate)
current comes out both sides, needs to be able to switch between

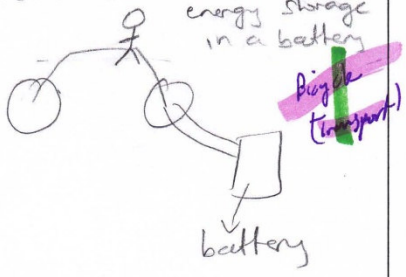
AS-PHI-A

Hand Crank



Bicycle

The person pedals the wheels and the motion leads to energy storage in a battery



Foot Pedal system

Perso taps his/her foot on a pedal and the motion is converted to electrical energy or something.

foot w/ pedal

Hamster Wheel?

But for humans that is. Like a giant wheel

human w/ hamster wheel

Pogo Stick

The mechanical vibrations converted to electrical energy.

human w/ pogo stick
(Input)

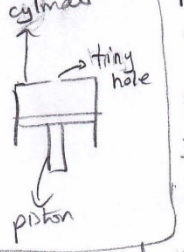
Rubbing hands together to create friction/heat, then using this heat to power up a device

heat → electrical energy.

heat w/ friction
(produce)

AS-PHS-A

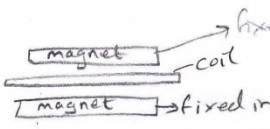
Pump / Piston cylinder



Rapid movement of the pump, can expel a highly pressurised stream of air through the hole, this pressurised air can then move a turbine or something

pressure w/ piston (Motion)

air of turbine (Input)



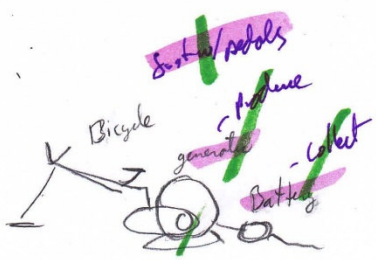
EC of the induction

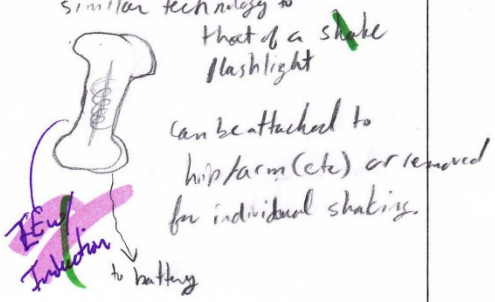
A person can move the coil in between the permanent magnets and that would cause a current to be induced in the coil. This current can then be harnessed.

hand w/ coil (Input A)

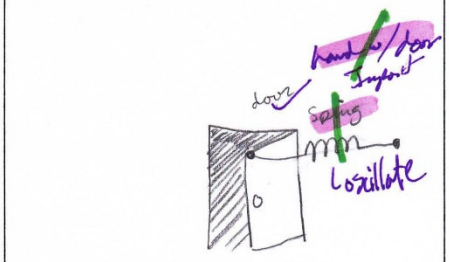
A6-PHI-A

jumping
walking



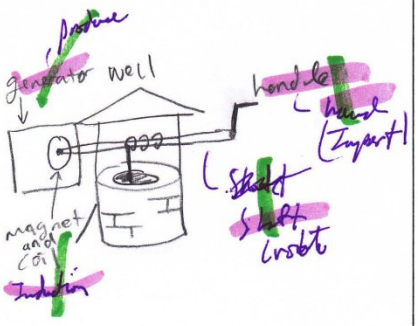


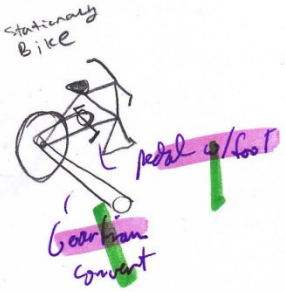
A7-PH-A



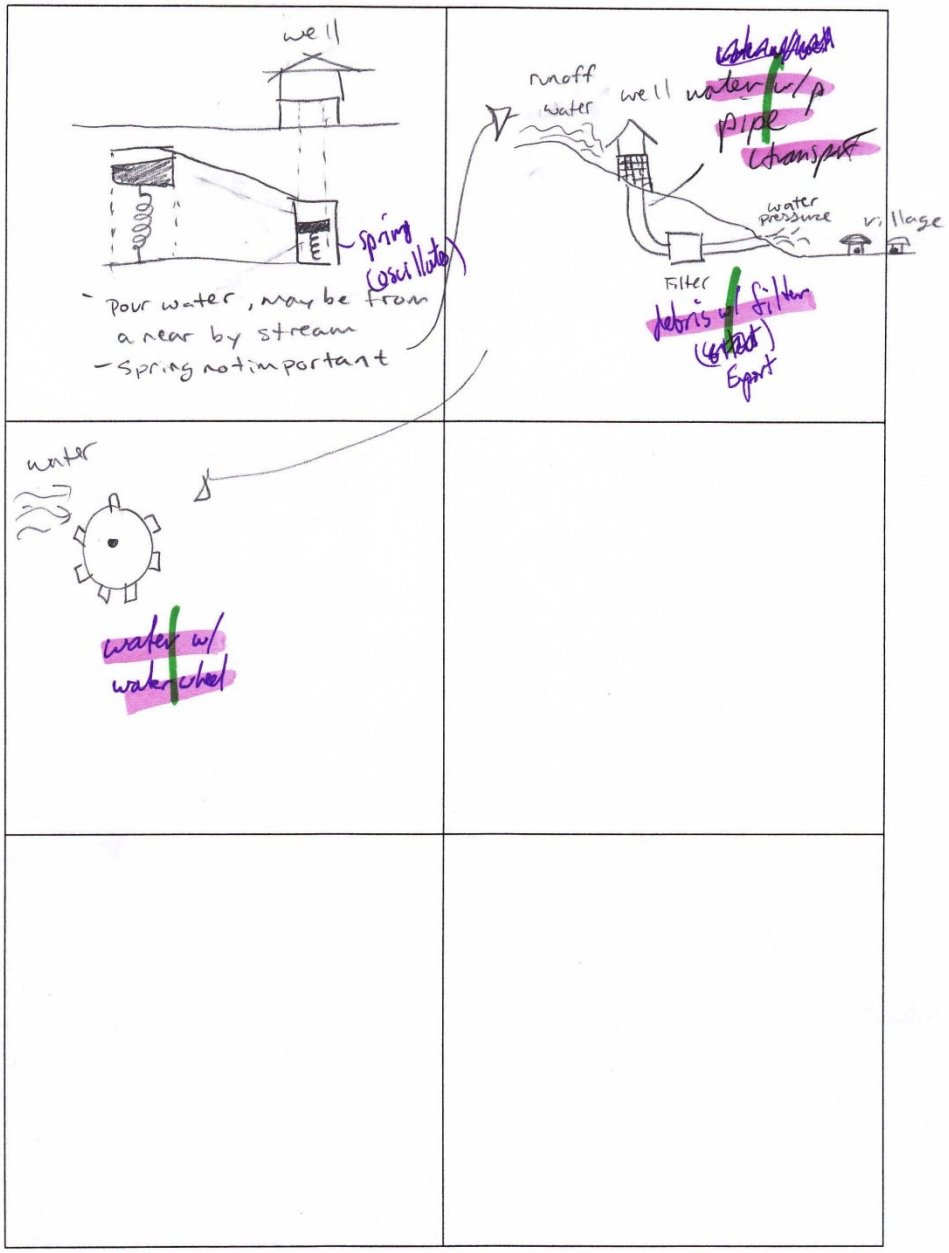
-spring connected to a clutch that doesn't allow spring to move back
-spring stores E

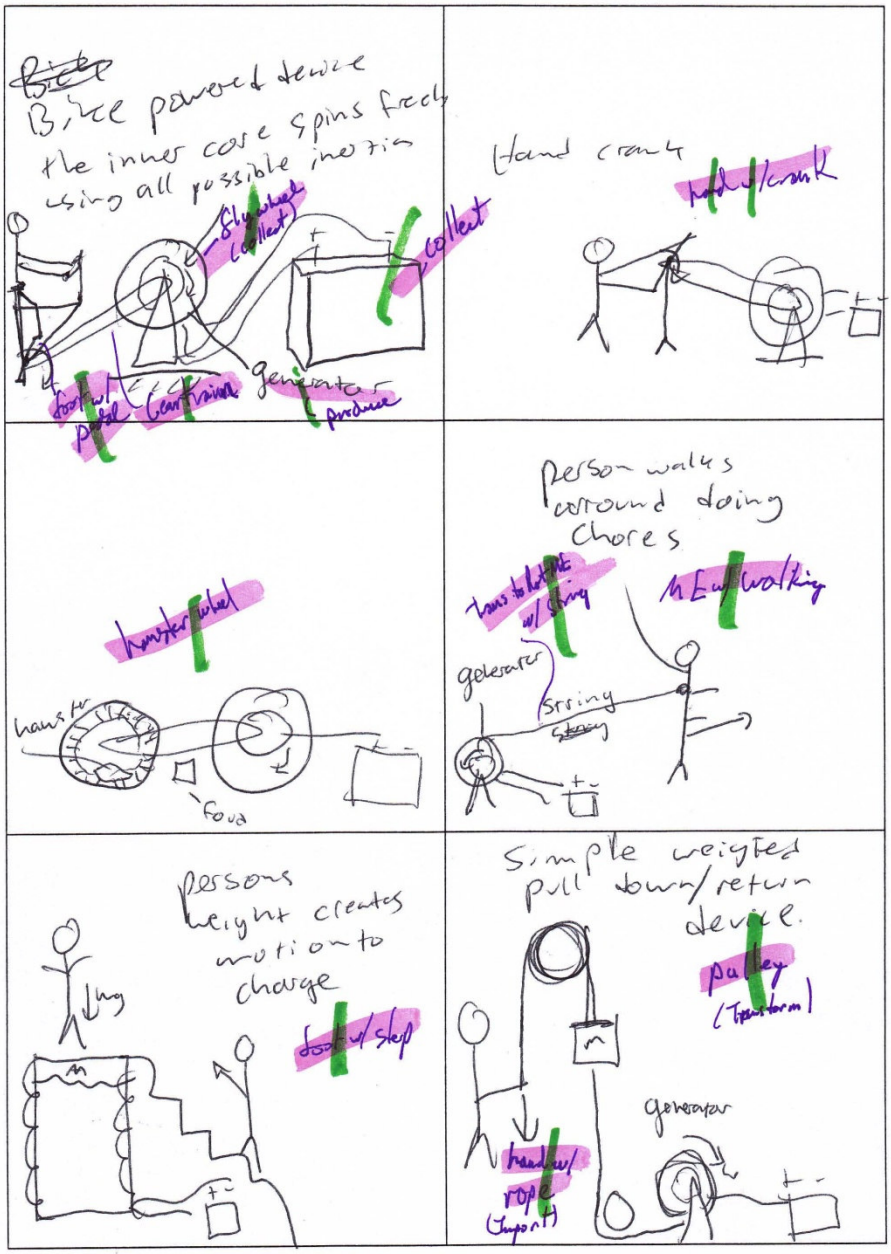
Transform



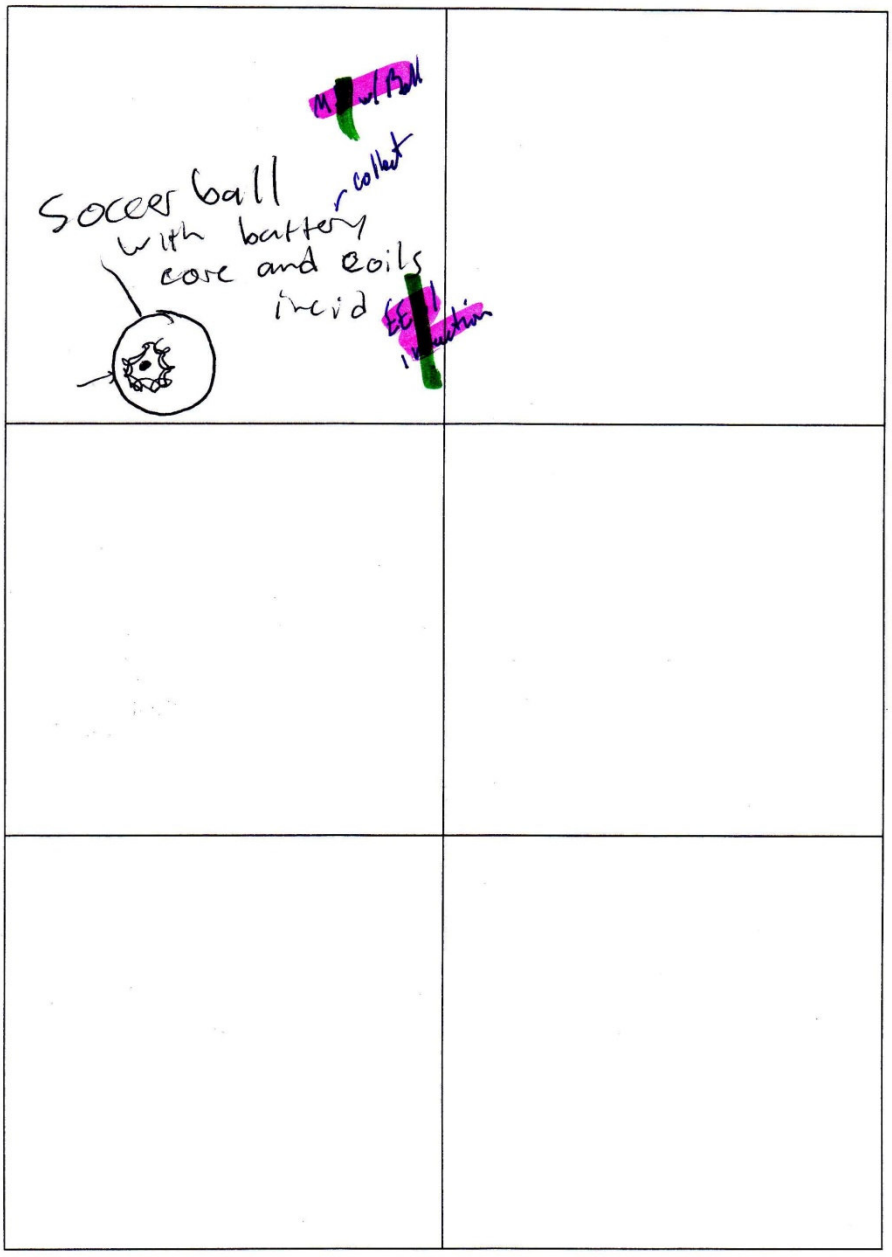


A7-PH3-A

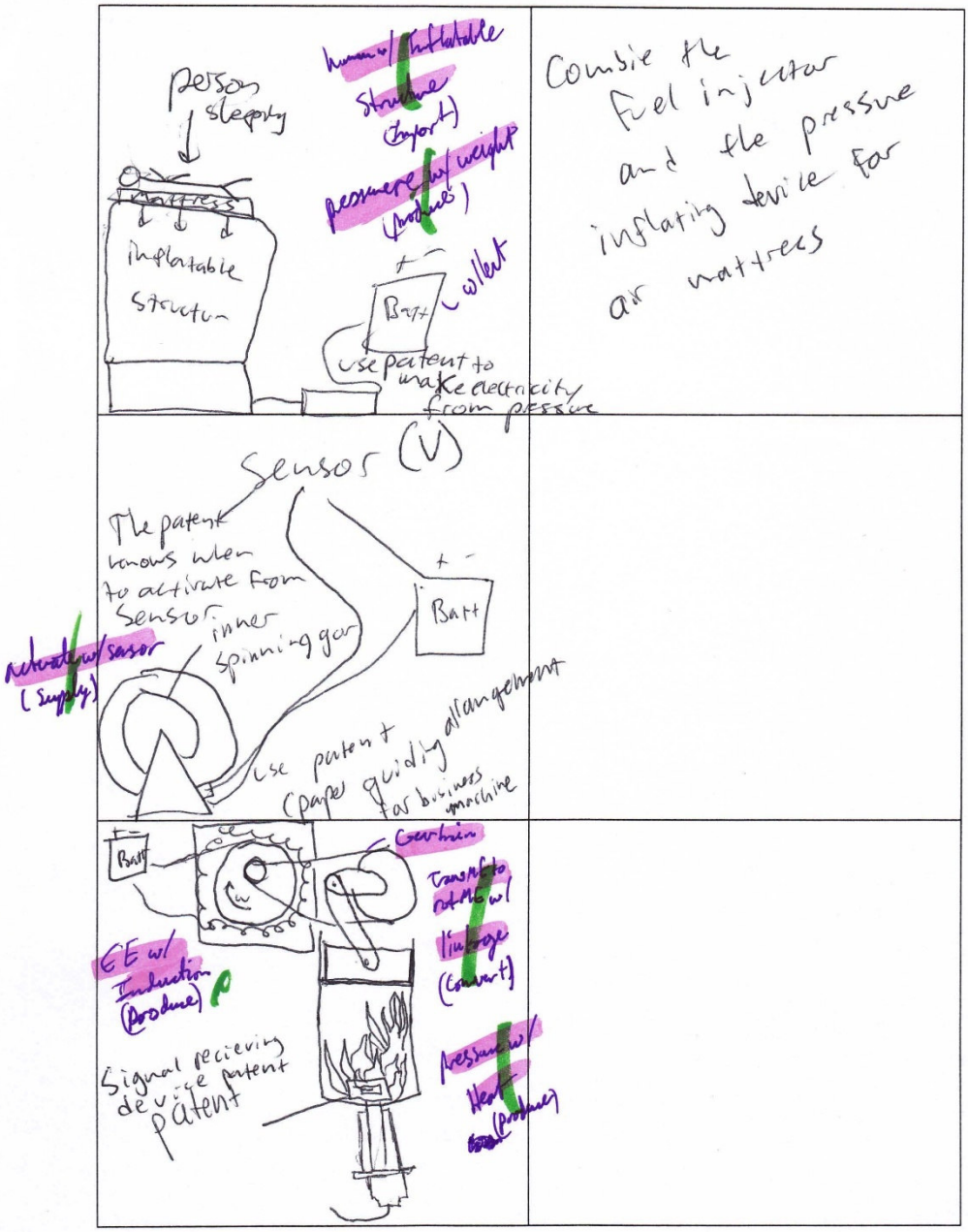

well
spring
(oscillator)
- pour water, may be from a near by stream
- spring not important
runoff water
well
water w/p
pipe
transport
water pressure
village
filter
debris
filter
(grit)
Export
water
water w/ water wheel


Bike powered device
the inner core spins freely using all possible inertia
Hand crank
Person walks around doing chores
Persons weight creates motion to charge
Simple weighted pull down/return device.
pulley
generator

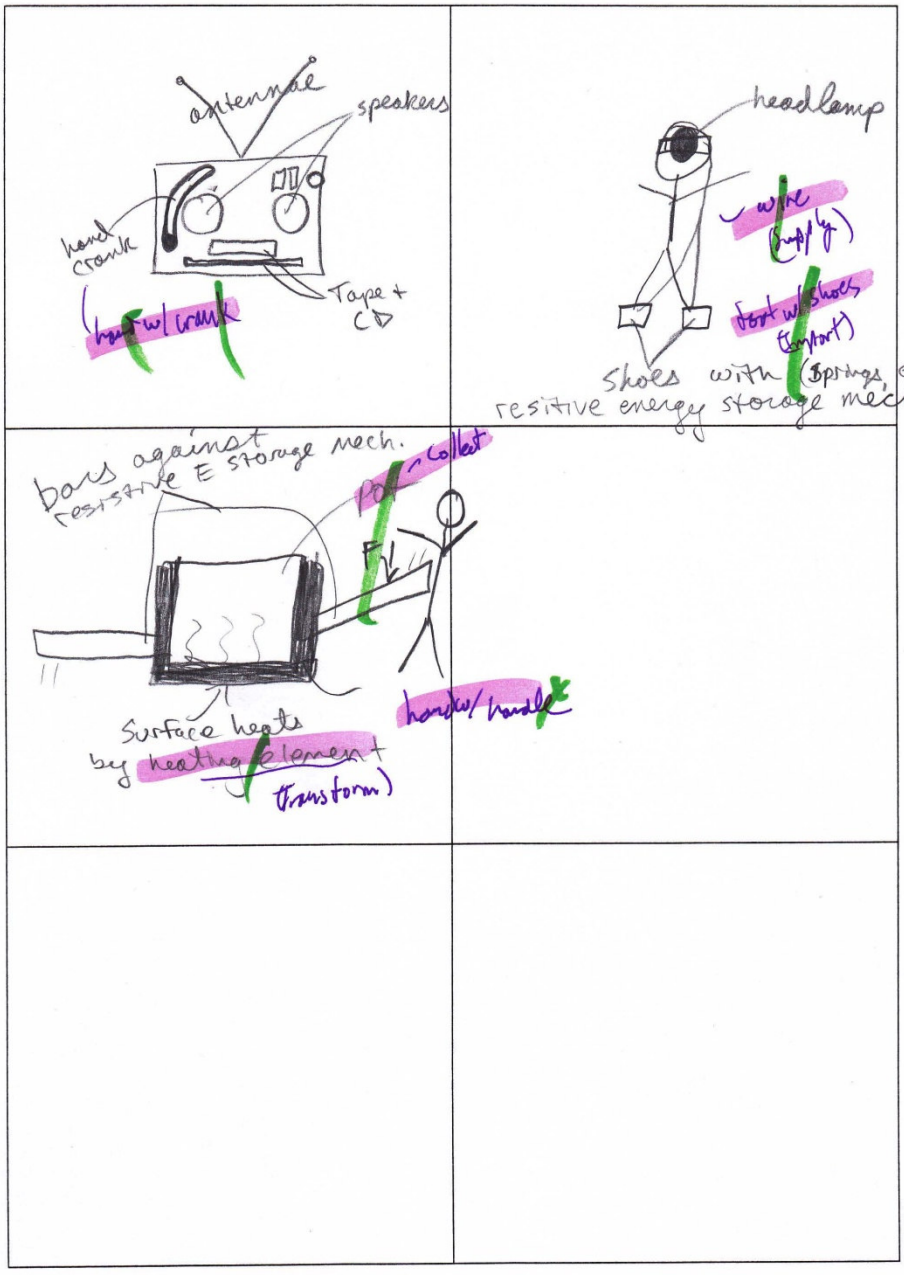
AS-PHI-B



AB-PH3-A

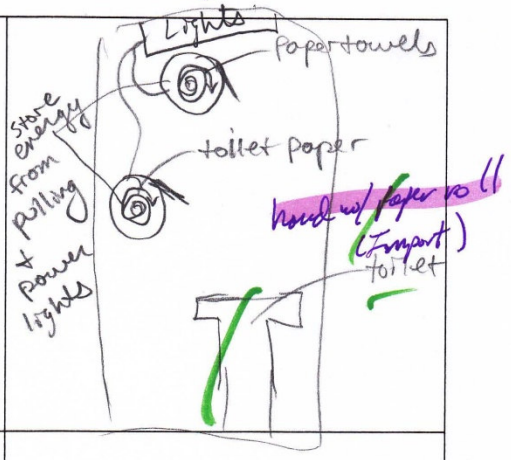

person sleeping
mattress
inflatable structure
Batt
use patent to make electricity from pressure
Consider the fuel injector and the pressure inflating device for air mattress
Sensor (V)
The patent knows when to activate from sensor
inner spinning gear
Batt
use patent (paper guiding alignment for business machine
Batt
Gearing
Signal recieving device patent
patent

A9-PHI-A


antennae
speakers
hand crank
Tape + CD
headlamp
wire (supply)
foot w/ shoes (comfort)
Shoes with (springs, etc.) resitive energy storage mech.
bars against resistive E storage mech.
pot
collect
handle w/ handle
Surface heats by heating element
transform)

A9-PHS-A

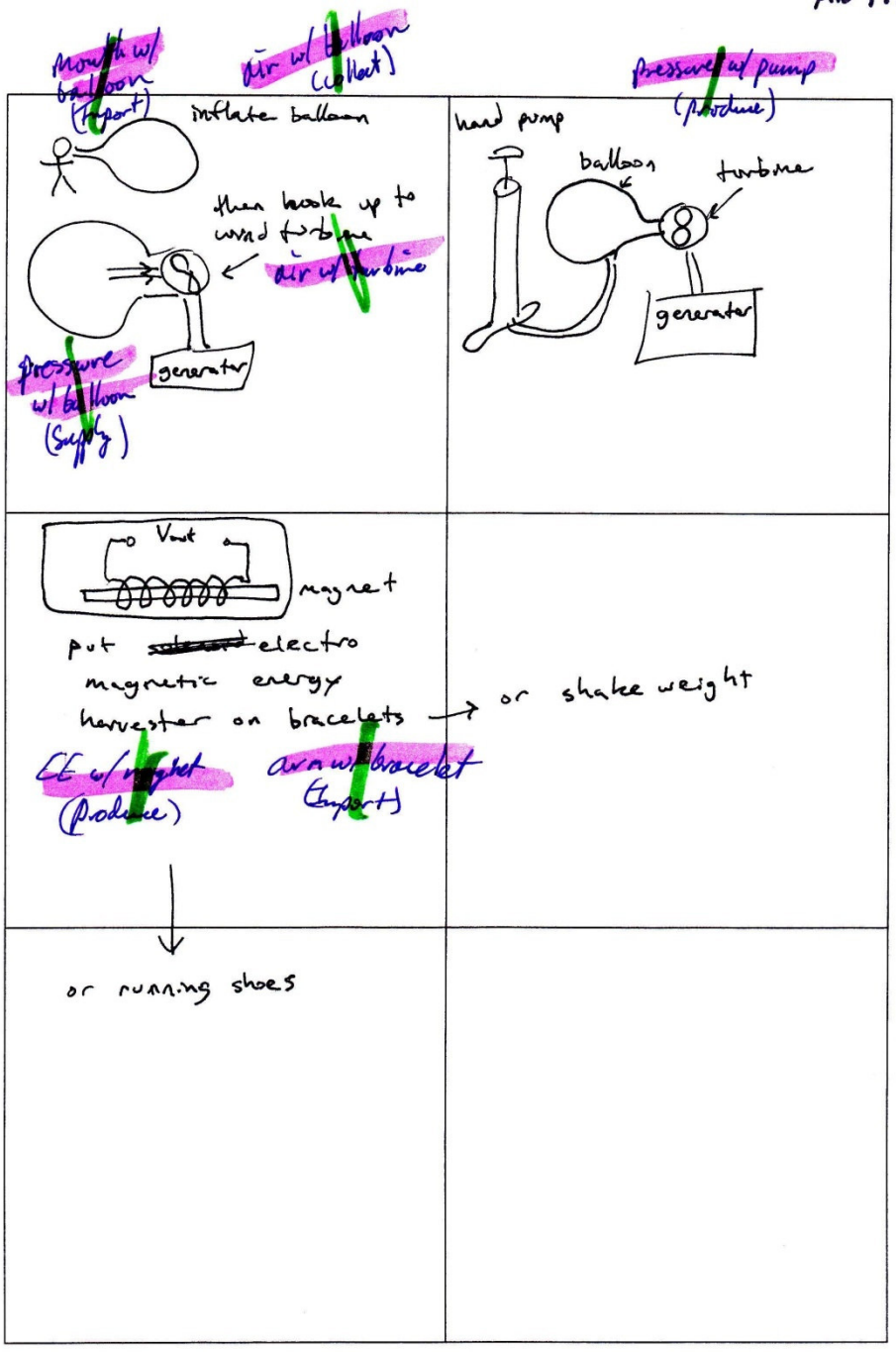
light w/ lights (Supply)

Use wireless comm. device w/ radio.	
Lights that only use as much power as is needed for brightness.	

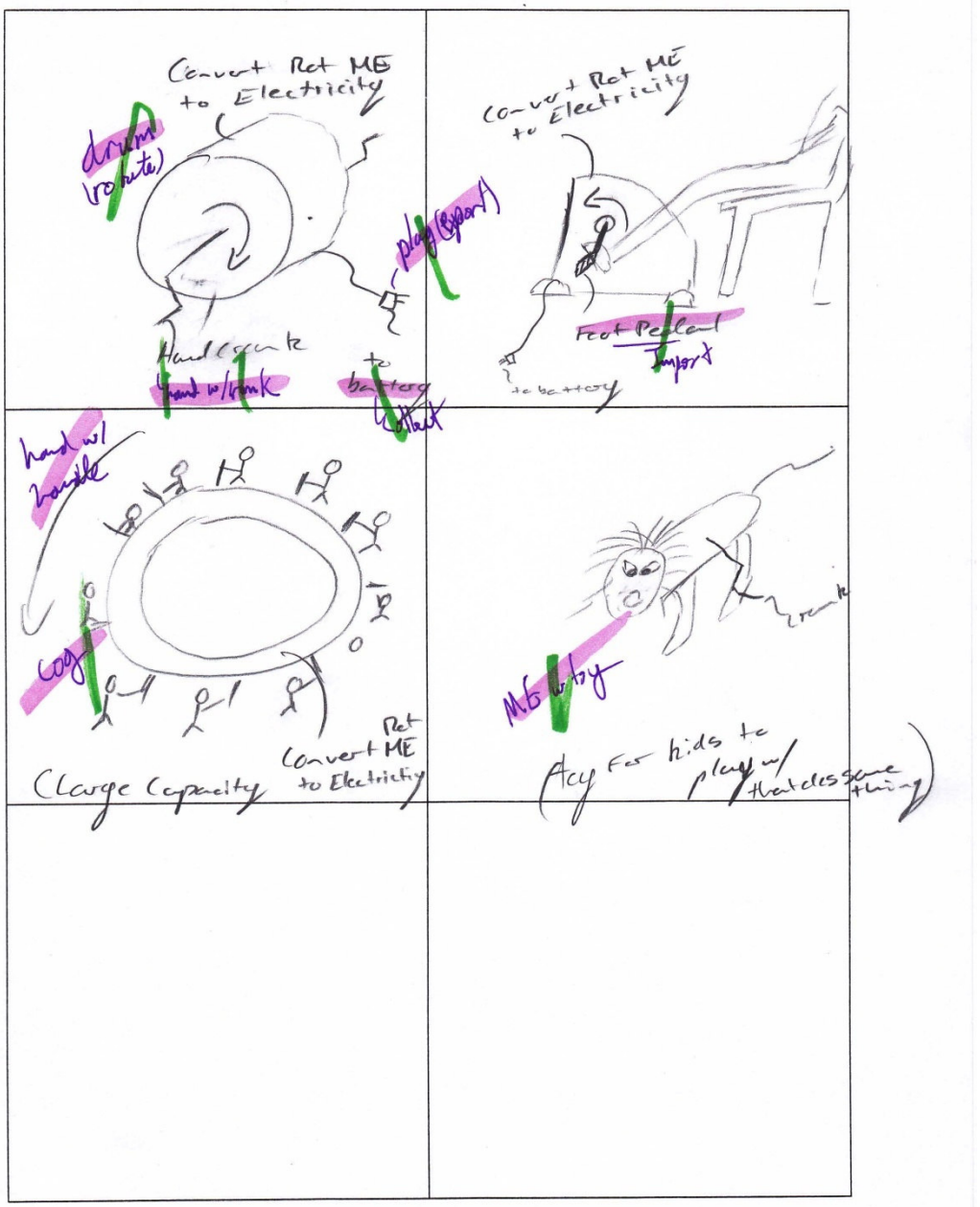
A10-PHI-A

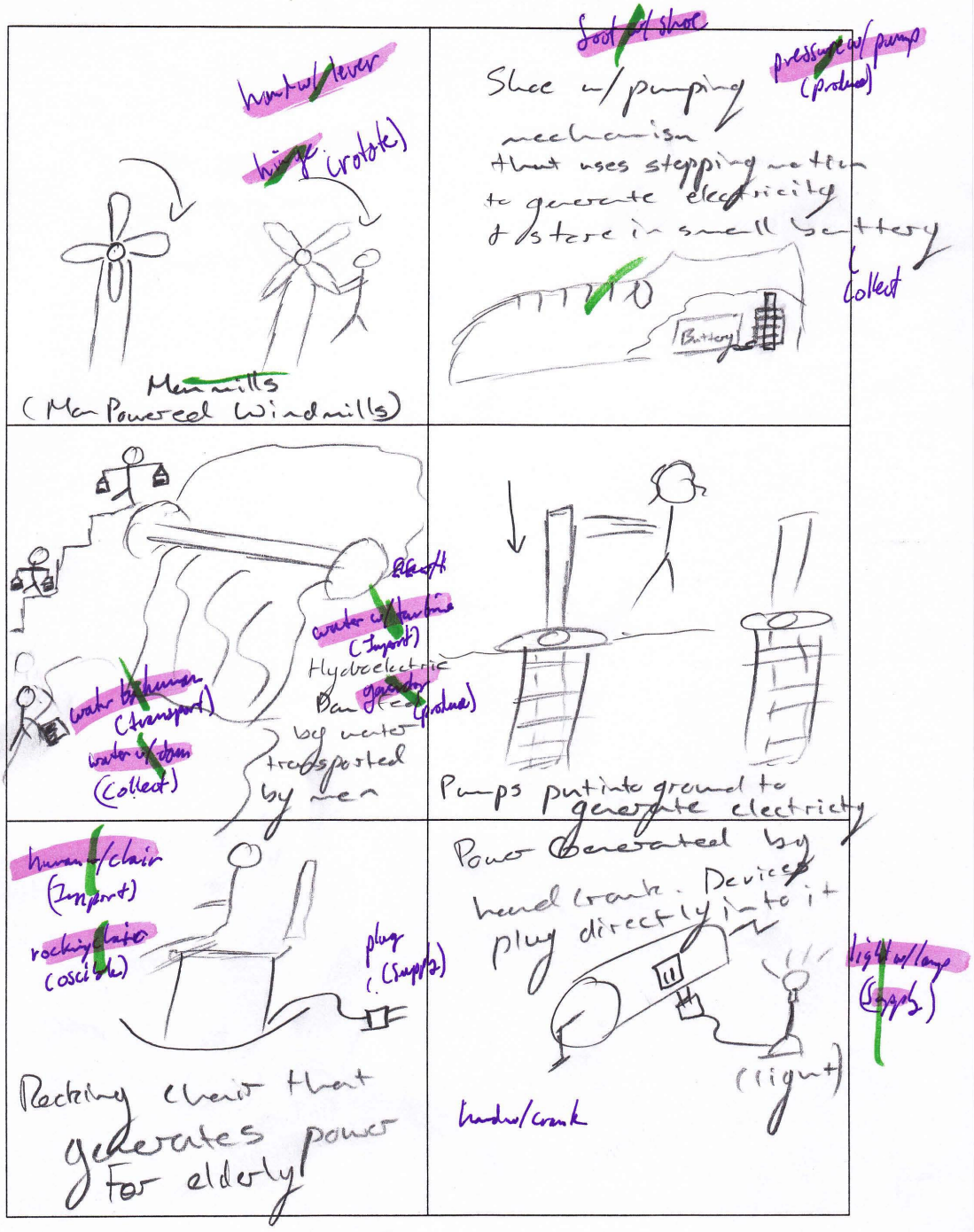
use generator hooked up to a bicycle that stores energy in a battery produce collect transport foot w/ pedals	stationary bike that runs a generator that stores energy in a battery
hand crank generator hand w/ crank	piezoelectric crystals attached to shoes produce bottom of shoe
piezoelectric crystals installed under doorway or high foot traffic area foot w/ floor	rowing machine that runs an electric generator rope handle spool wind up spring shaft (rotate) Gen

A10-PH3-A


mouth w/ balloon (input)
air w/ balloon (output)
inflate balloon
then hook up to wind turbine
air w/ turbine
generator
pressure w/ balloon (supply)
hand pump
pressure w/ pump (produce)
balloon
turbine
generator
Vout
magnet
put electro magnetic energy harvester on bracelets
or shake weight
EE w/ magnet (produce)
arm w/ bracelet (input)
or running shoes

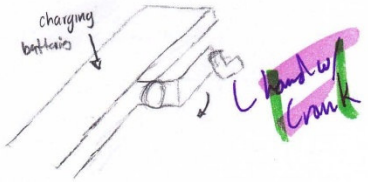
All-Phi-A-


Convert Rot ME to Electricity
drive (route)
plug (support)
Hand crank
to battery
Convert Rot ME to Electricity
Foot Pedal
support
to battery
hand w/ handle
cog
Charge Capacity
Convert Rot ME to Electricity
ME w/ toy
(Acy For kids to play w/ that does some thing)


human w/ lever
hinge (rotate)
Men mills
(Man Powered Windmills)
foot w/ shoe
pressure w/ pump
(produce)
Shoe w/ pumping mechanism that uses stepping motion to generate electricity & store in small battery
Battery
collect
Electricity
water w/ turbine
(Import)
Hydroelectric Dam (Generate)
water transport
(Transport)
water storage
(Collect)
big water transported by men
Pumps put into ground to generate electricity
human w/ chair
(Import)
rocking chair
(oscillate)
plug
(Supply)
Rocking chair that generates power for elderly
Power generated by hand crank. Devices plug directly into it
light w/ lamp
(Supply)
(light)
human w/ crank

A12-PH1-A

a manual turn crank that charges a ~~battery~~ (toilet) for simple tasks



(crank)

each household could own a bike that directly charges small electronics for everyday use.

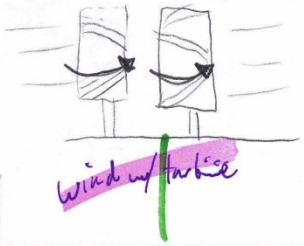
bike (transport)

For elderly people can have cane's with lights but powered by a crank shaft

(cane)

A12-PH3-A

Implement wind turbines on top of buildings to make them self sustaining.



AB-PHI-A

stationary bicycle that turns a generator	~~treadmill connected to~~ Chemical process to harvest trash and human waste to harness energy.
Make all villagers wear solar panel T-Shirts and hats ~~with solar~~ connected to batteries	Faraday's Law shake and generate power system
gear and crank system that can be run by human to create power.	shoes that capture energy with springs

Harness energy from a jump-rope. ME w/ jump rope	Create a device that can ~~perform~~ utilize ATP ~~and~~ and then feed it ~~fresh~~ food ppl have left-over like rotten food.

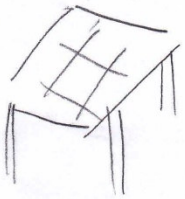
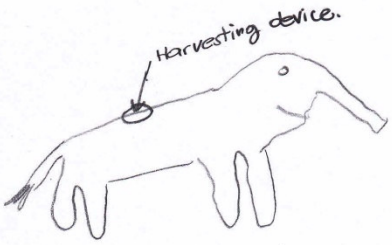
A13-P13-A

shoes studded w/ solar powered pressure transducers

Air pump/foot pump to turn fan & power battery

feed paper into machine continously and use machine gears to turn generator and power battery.

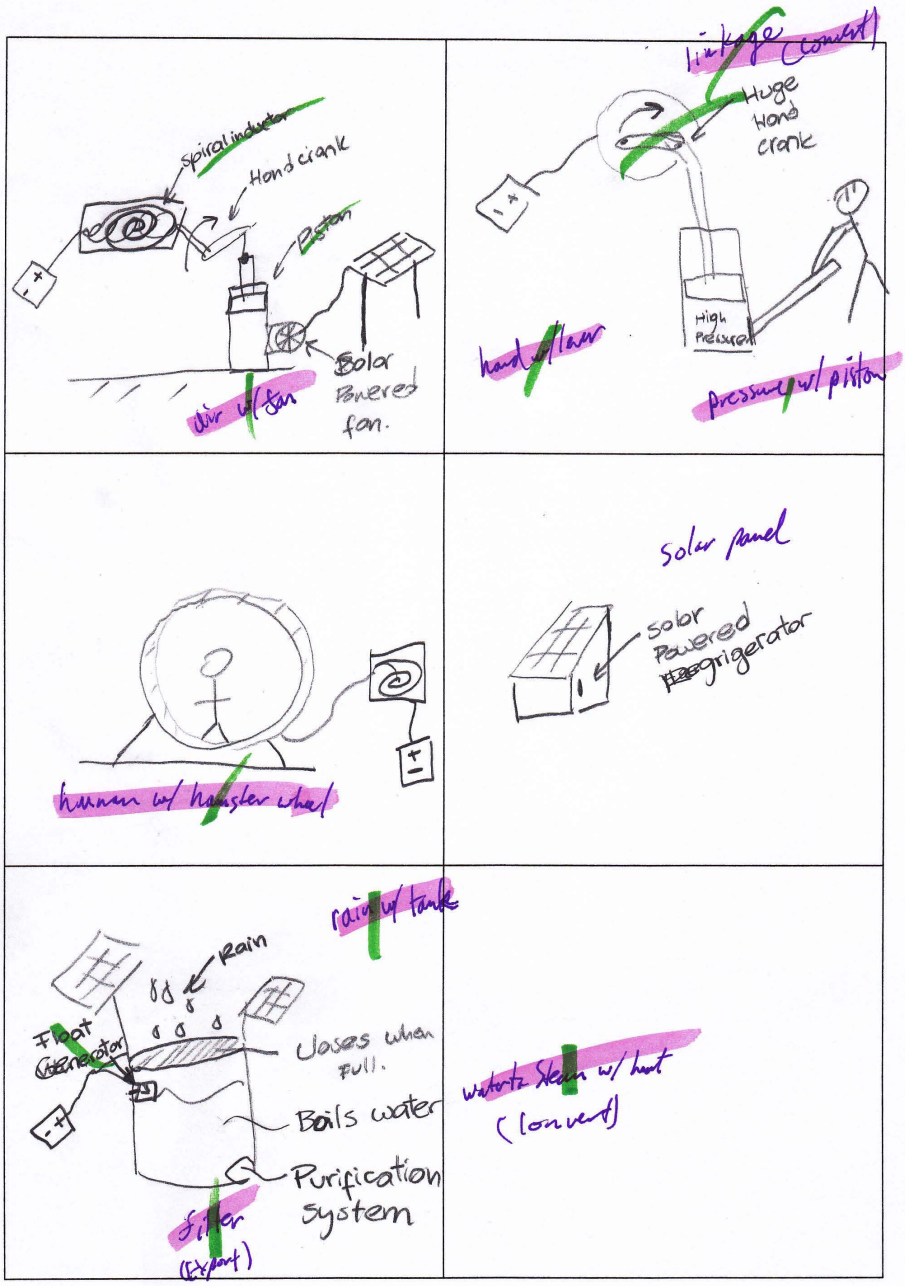
A14-PHI-A

Solar power 	Geothermal
Hand cranks, Footpedals flywheels	Biofuel.
Vibrations. 	Create a mini-power plant

AK-PHI-D

Sell oil/kerosene and make electricity from that.	

A14-PH3-A


Spiral inductor
Hand crank
Piston
Solar Powered fan.
air w/ fan
linkage (connect)
Huge Hand crank
High Pressure
hand w/ lever
pressure w/ piston
human w/ hamster wheel
Solar panel
solar Powered refrigerator
rain
rain w/ tanks
Float
Generator
Loses when full.
Boils water
Purification system
filter
(Export)
water to steam w/ heat
(convert)

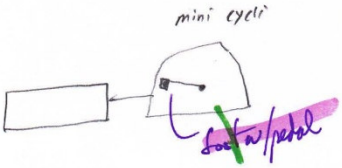
AIS-PHI-A

Hand crank radio

Hand crank battery charger

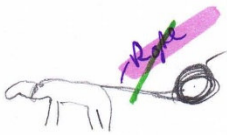
A small mini cycle that charges batteries during operation

mini cycle hooked up to battery charger

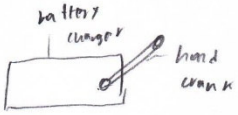


use animales to harves their energy.

have animales hooked up to ropes that can spin a small generator



generator also charges batteries



Solar panels that can charge batteries.

be able to store solar energy during day so it can be used at night.

use vibration energy

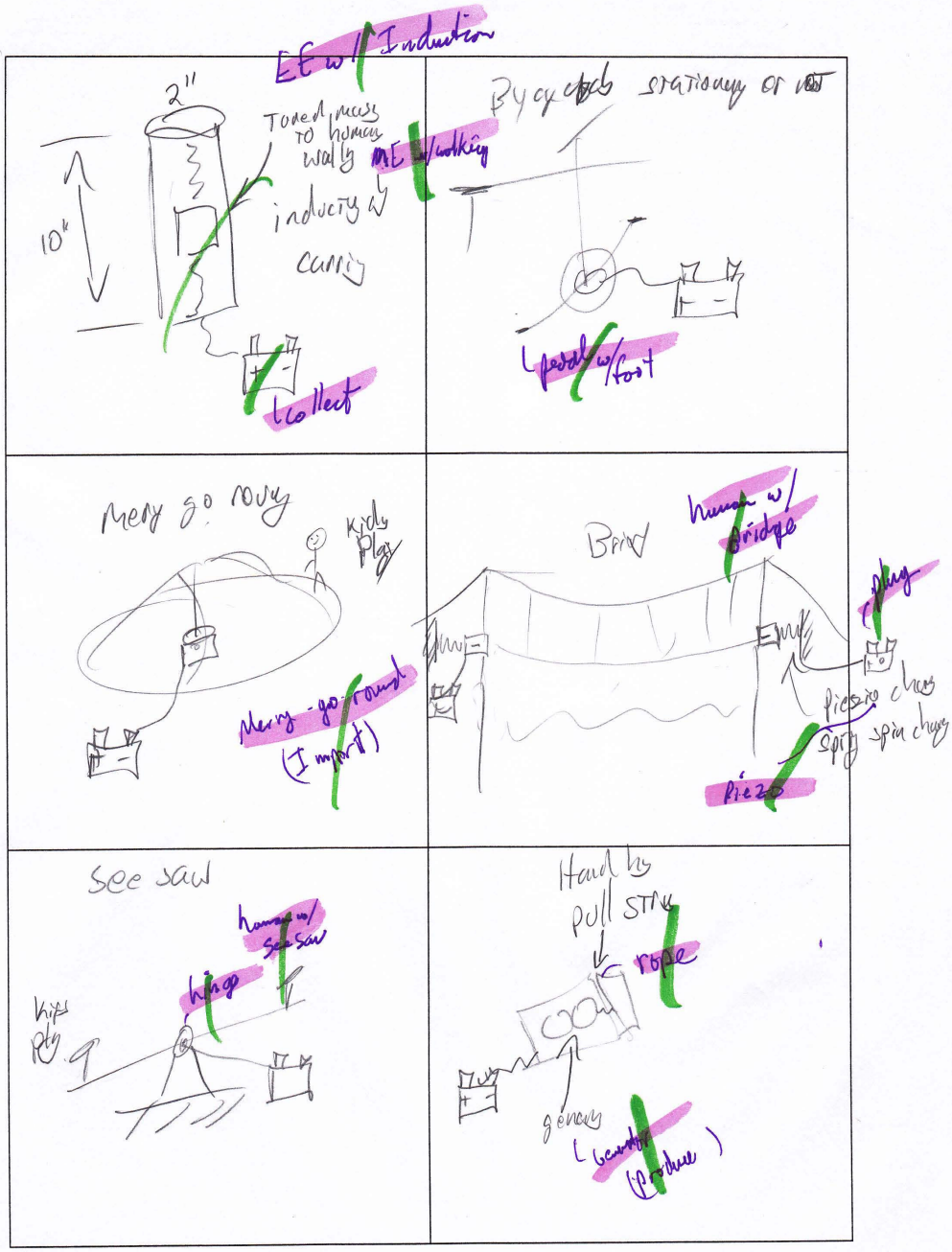
A15-PH1-B

stationar bycylle to ~ Seat w/ pedals gain energy from motion and harvest that energy	

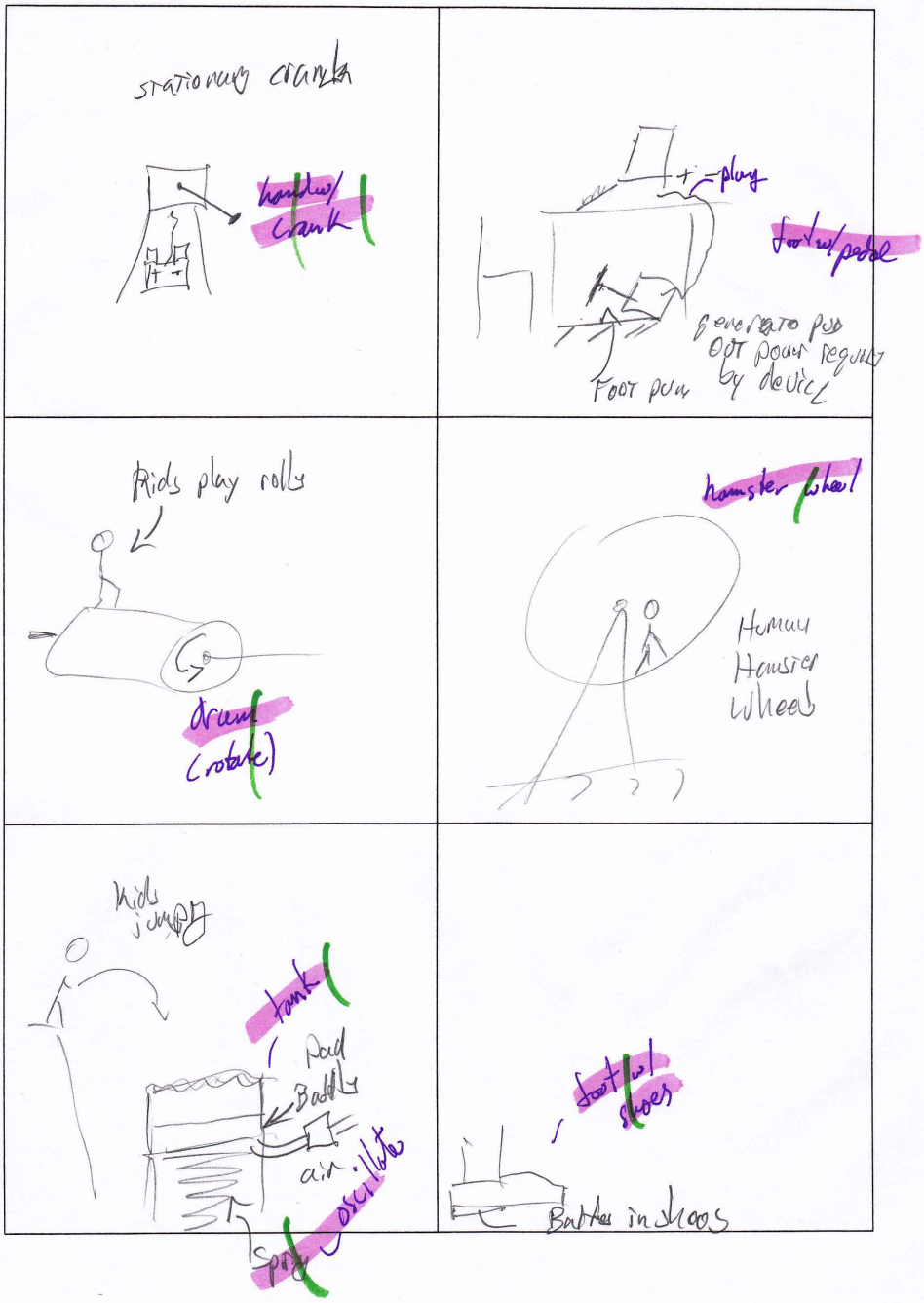
A15-PH9-A

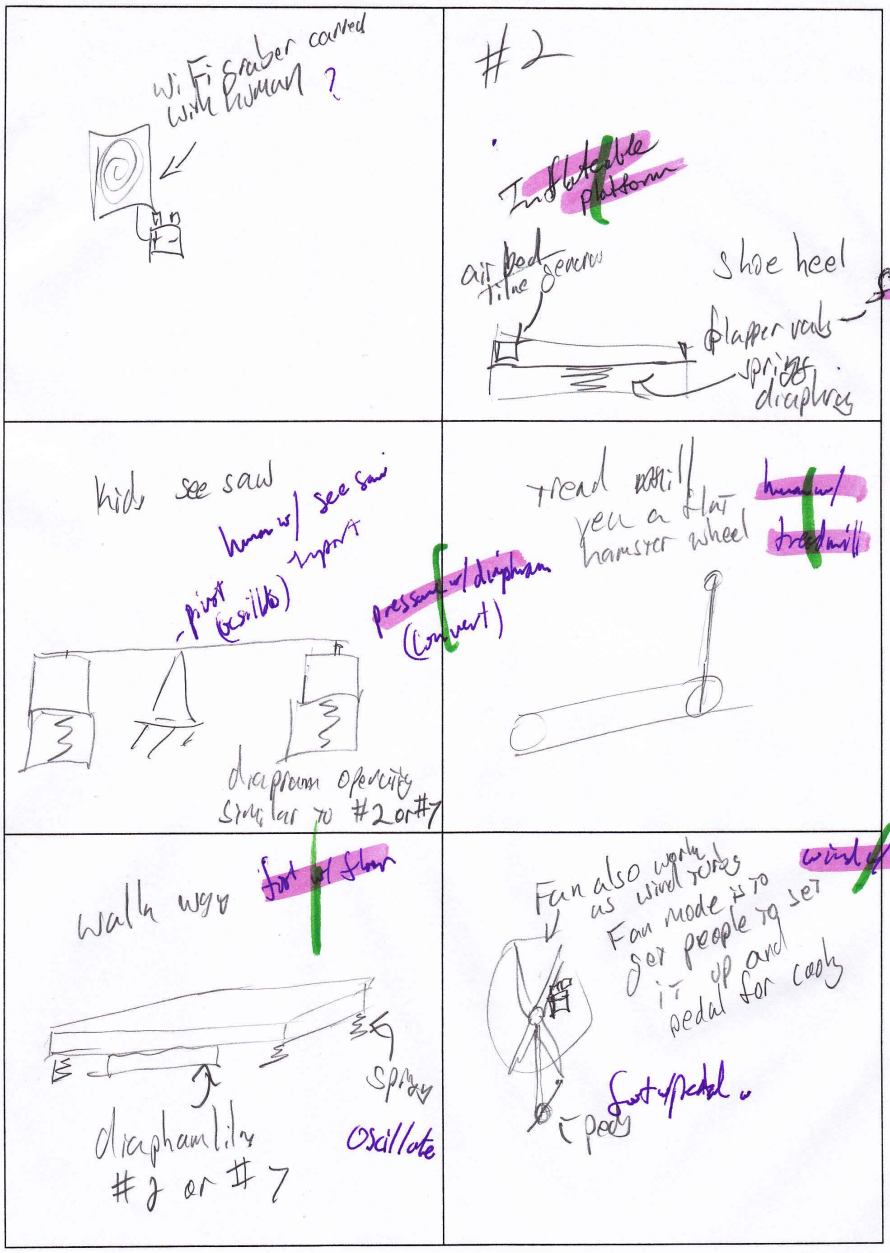
the fuel injector and air mattress helped me come up w/ this idea

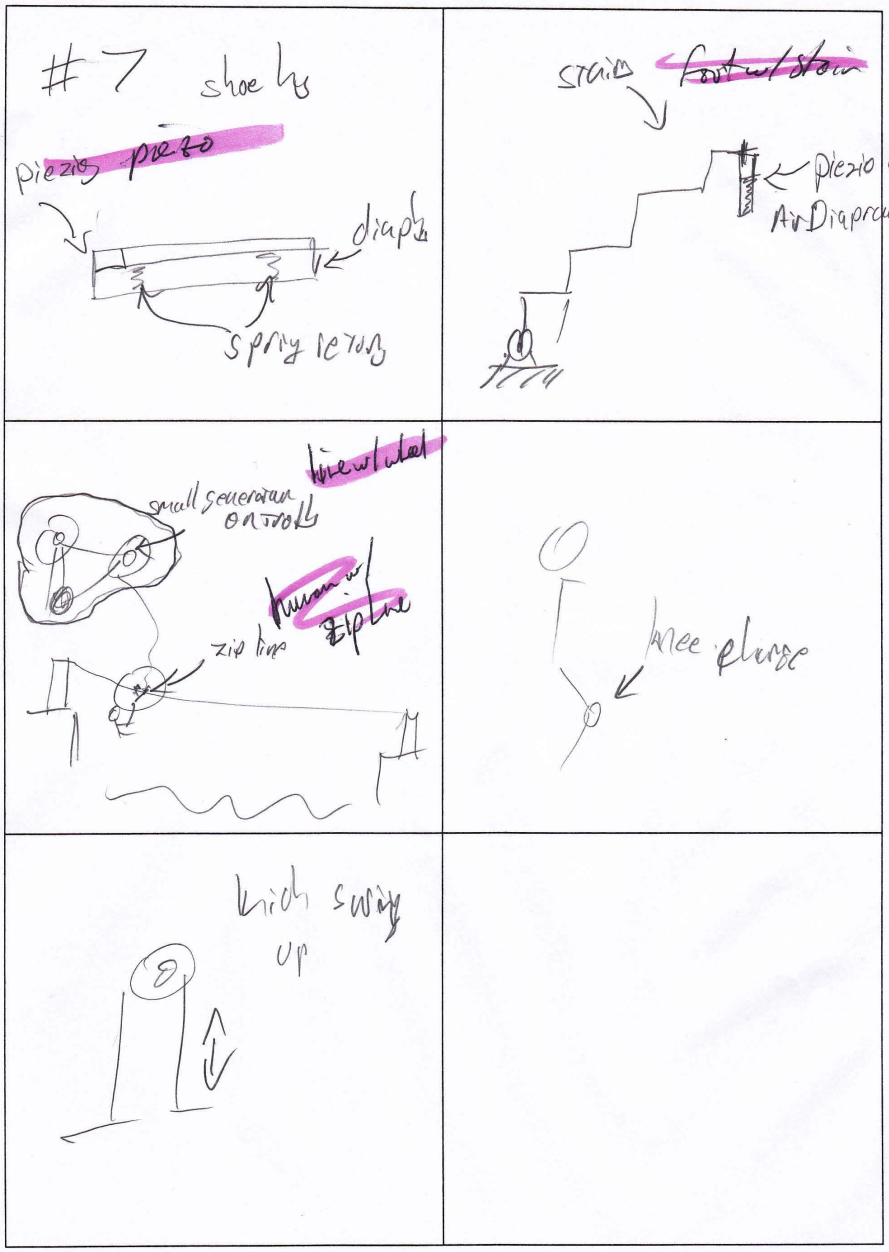
A series of valves can be used that can allow	A Sterling engine can be used by heating up air and cooling it causing a pressure difference Pressure w/ piston ΔT w/ Sterling Engine (Convert)  this can provide energy w/o fuel it just needs air. can charge batteries
A wind mill can be used that aligns itself w/ the wind to capture energy and use it to charge batteries. air w/ windmill	


EE w/ Induction
2"
10"
Tuned mass to human walk
induct
coil
collect
By cycle stationary or not
pedal w/ foot
Merry go rouy
kids Play
Merry-go-round
(Impact)
Bridge
human w/ bridge
Piezo chips
spray spin chips
Piezo
See saw
human w/ see saw
hinge
kids Play
Hand by pull string
rope
gearing
(Crank)
(Pulley)

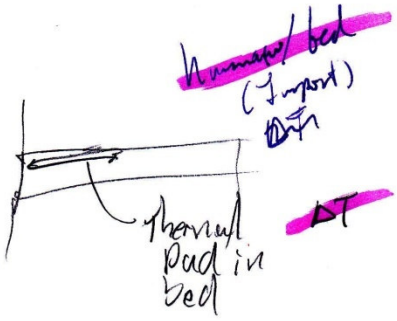
A16-PHI-B

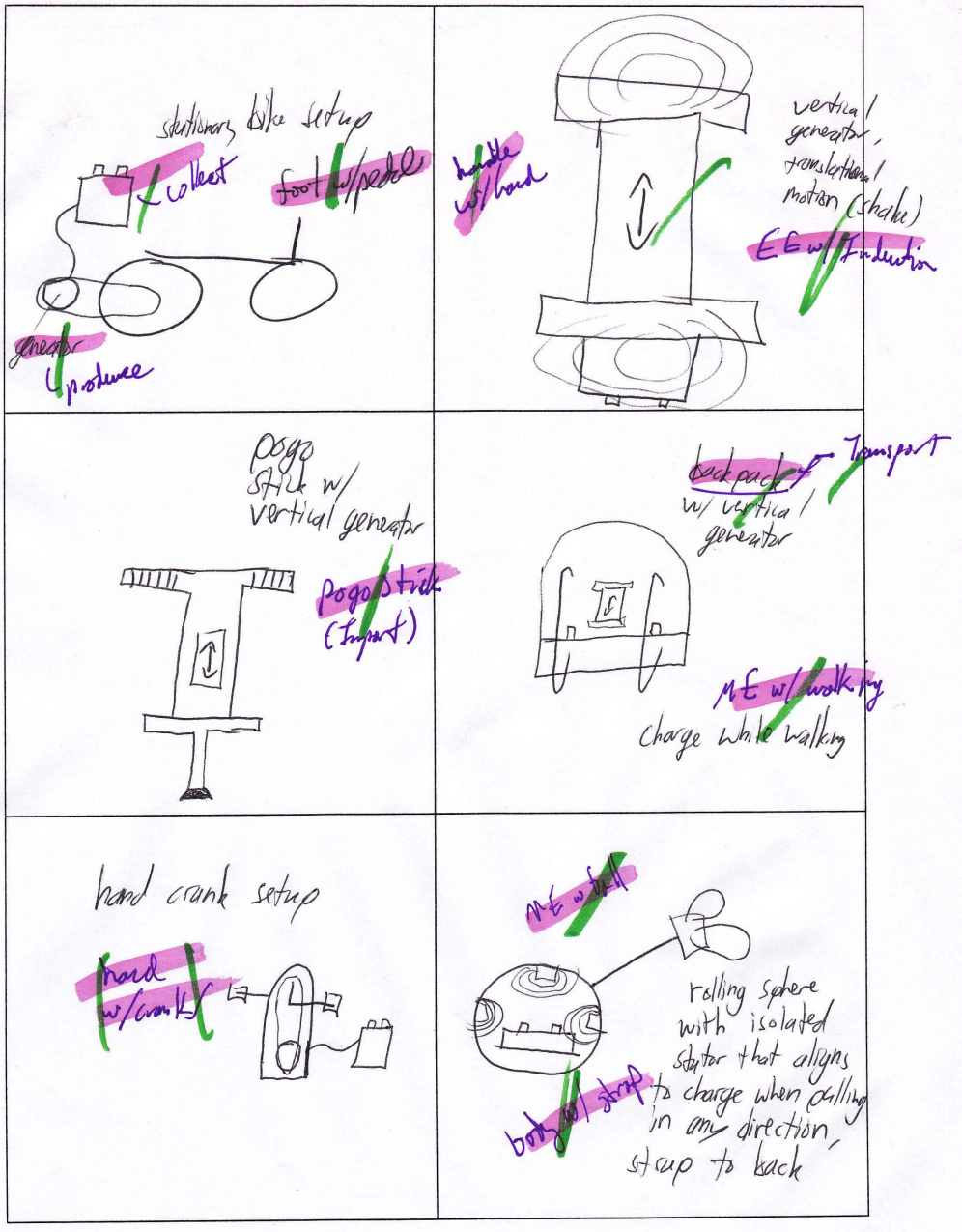

stationary cranks
handle/ crank
+ -plug
foot rest/pedal
Foot pump
Kids play rolls
drum (rotate)
hamster wheel
Human Hamster Wheel
Kids jump
tank
Pedal
Baffle
air
oscillate
foot w/ shoes
Batter in shoes


wi Fi Saber carved with human ?
#2
Inflatable platform
air pod time sensor
shoe heel
flapper valve
springs diaphragm
kids see saw
human w/ see saw
pivot (scrolls)
input
pressure/diaphragm (convert)
diaphragm operates similar to #2 or #7
treadmill
you a hamster wheel
human/ treadmill
walk ways
foot of flow
spring
diaphragm like #2 or #7
oscillate
Fan also work as wind power
Fan mode is to get people to set it up and pedal for cooling
wind of fan
pedal
foot/pedal

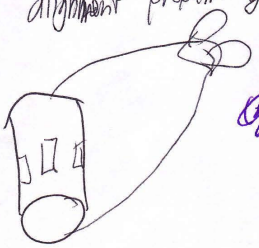
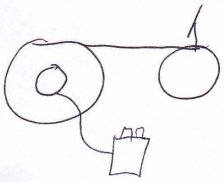

#7 shoe
piezo
diaph
spring rods
stairs
Piezo or
Air Diaphragm generator
small generator on track
zip line
knee plunge
kick swing up

AIV. PHS. B

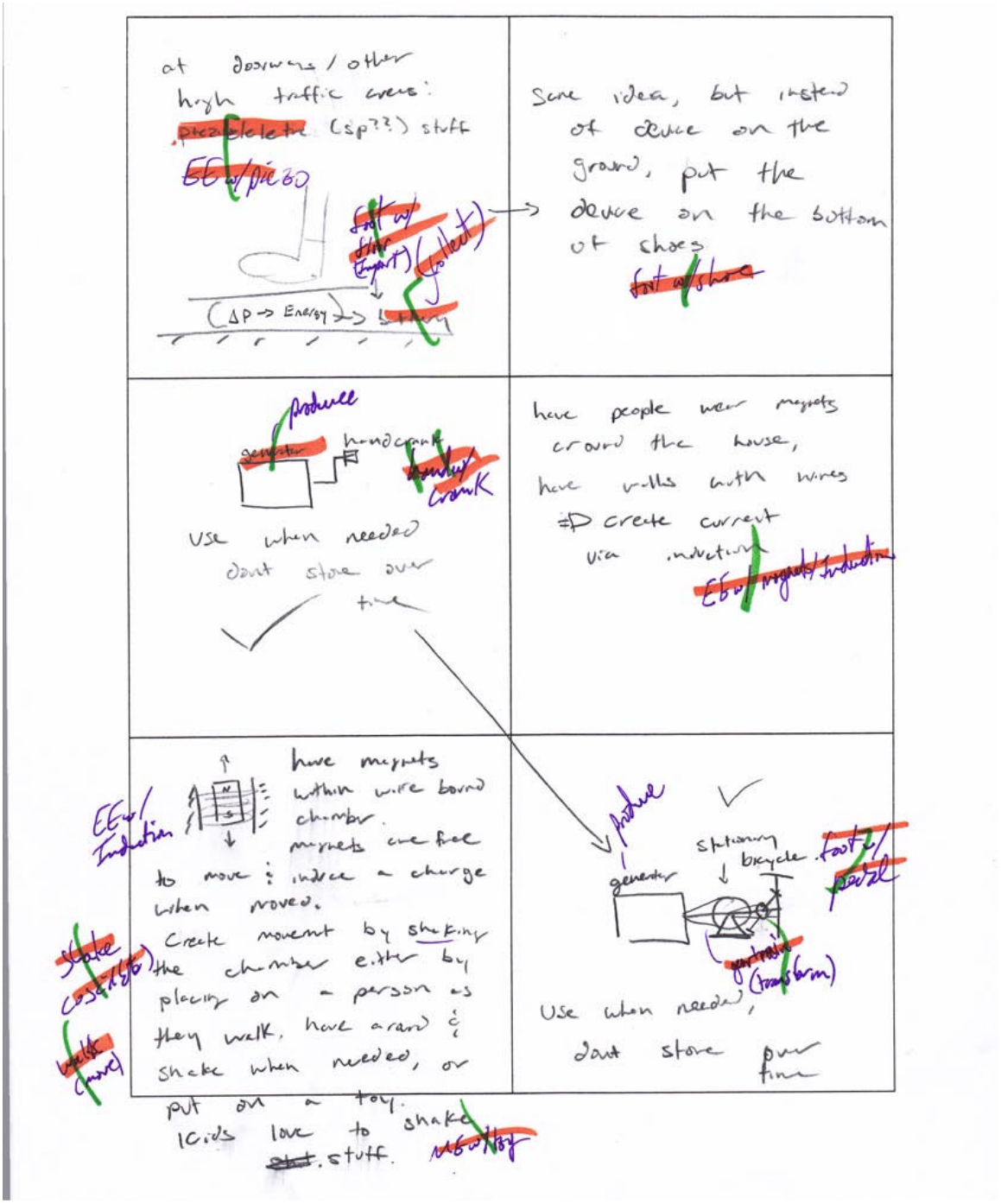



stationary bike setup
collect
foot w/pedals
generate
produce
handle w/ hand
vertical generator, translational motion (shake)
EE w/ Induction
pogo stick w/ vertical generator
Pogo Stick (Input)
backpack
Transport
w/ vertical generator
ME w/ walking
charge while walking
hand crank setup
hand w/crank
ME w/ wheel
rolling sphere with isolated stator that aligns to charge when rolling in any direction, strap to back
body/belt strap

1-PHS

battery charged by multiple units (any idea) by wirelessly broadcasting signal to excite circuit for inductance EE from RF / Antenna (current)	pulled cylinder (eliminate alignment problem geometrically)  Al w/ rope (rotate)
Transport bicycle idea: generator mounted to wheel instead of external 	

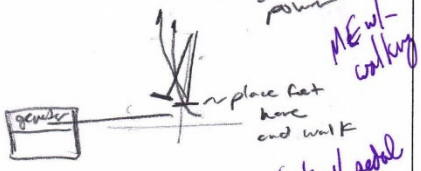
GROUP B


at doorways / other high traffic areas:
piezoelectric (sp??) stuff
EE w/ piezo
foot w/ shoe
(ΔP → Energy → Storage)
Same idea, but instead of device on the ground, put the device on the bottom of shoes
foot w/ shoe
produce
generator
handcrank
crank
Use when needed
don't store over time
have people wear magnets around the house,
have walls with wires
⇒ create current via induction
EE w/ magnets/induction
have magnets within wire bound chamber.
magnets are free to move: induce a charge when moved.
Create movement by shaking the chamber either by placing on a person as they walk, have a crank & shake when needed, or put on a toy.
Kids love to shake stuff.
EE w/ Induction
shake
crank
walk
produce
generator
stationary bicycle
foot / pedal
gearing (transform)
Use when needed,
don't store over time

BI-PHS-A

Instead of Stationary bicycle...

much like an elliptical machine to generate power

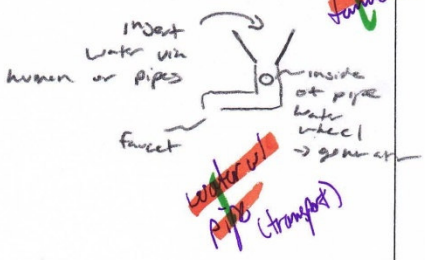


ME w/ walking

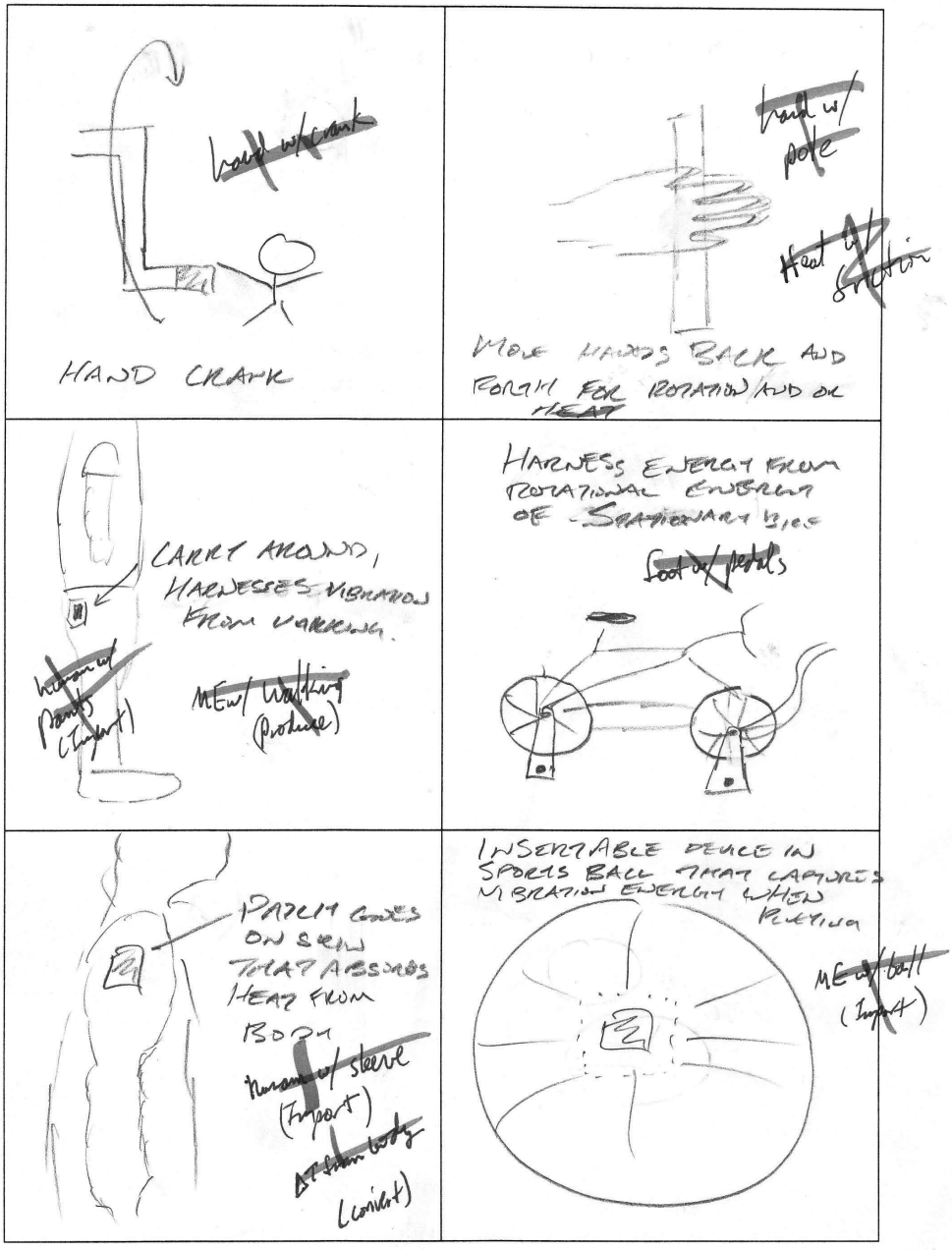
foot w/ pedal

In bathroom...

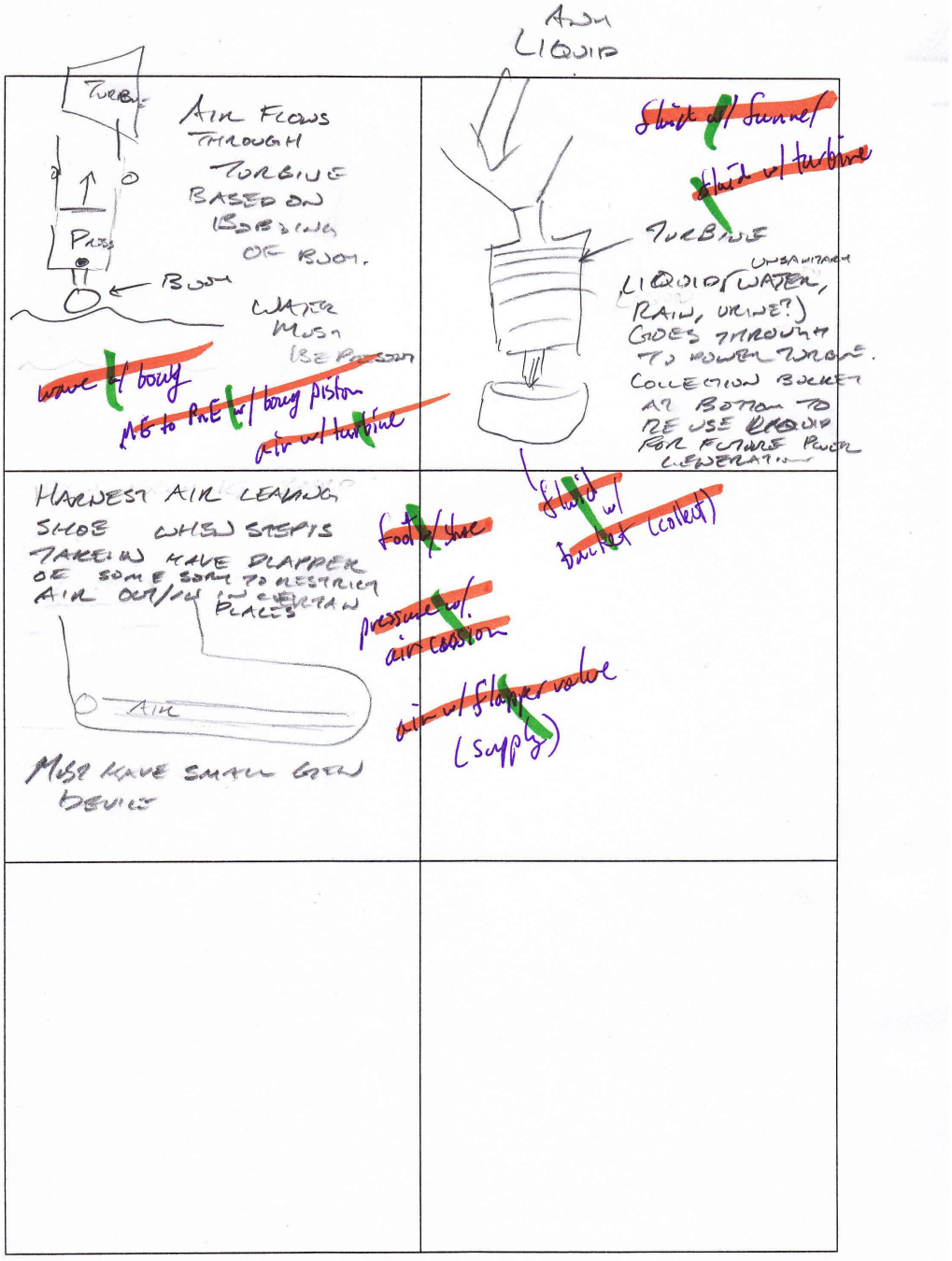
water w/ pressure



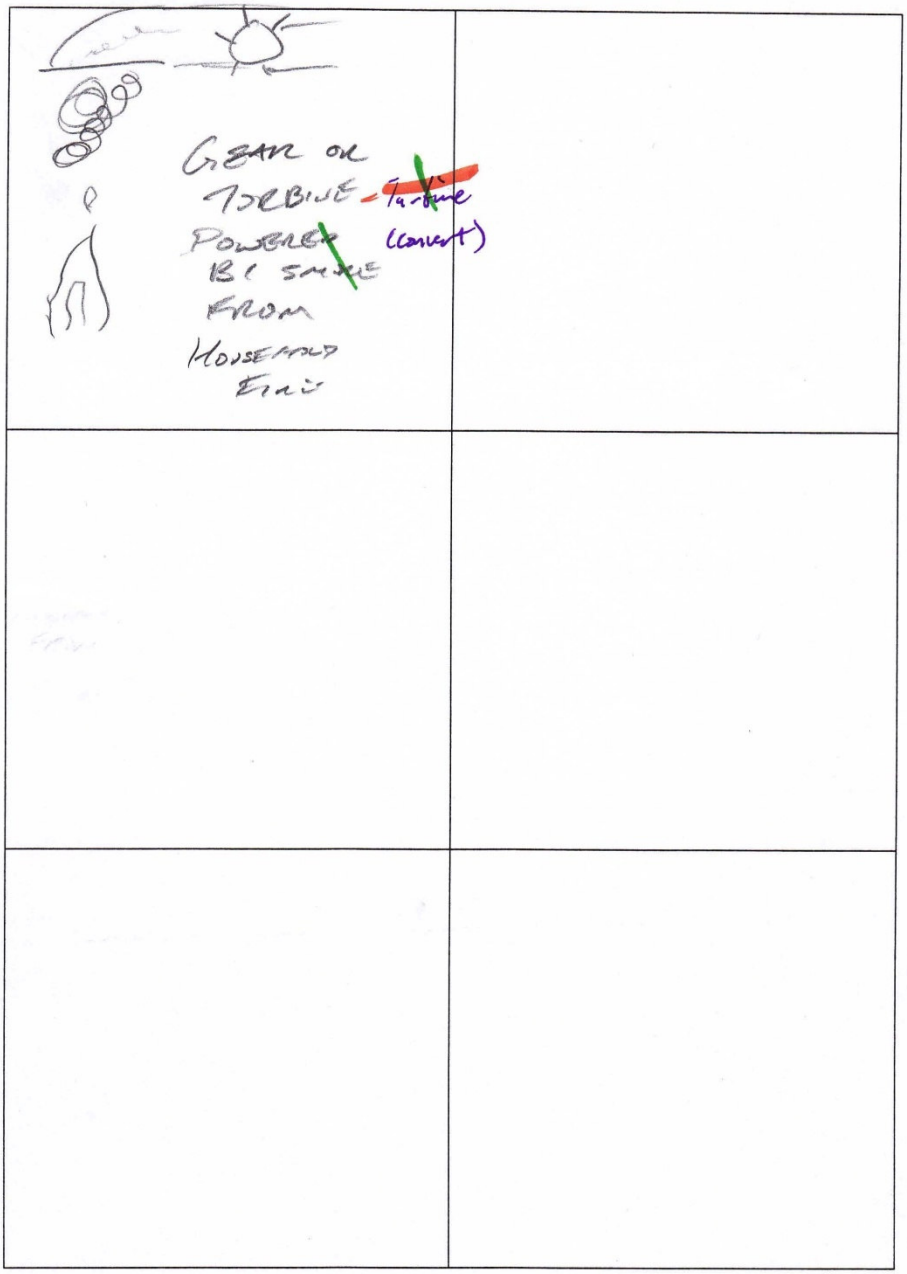
water w/ PE (transport)


HAND CRANK
MORE HANDS BACK AND FORTH FOR ROTATION AND OR HEAT
CARRY AROUND, HARNESSES VIBRATIONS FROM WALKING.
HARNESS ENERGY FROM ROTATIONAL ENERGY OF STATIONARY BIKE
PATCH GOES ON SKIN THAT ASSORBS HEAT FROM BODY
INSERTABLE DEVICE IN SPORTS BALL THAT CAPTURES VIBRATION ENERGY WHEN PLAYING

B2-PHS-A


AIR FLOWS THROUGH TURBINE BASED ON BLOWING OF BUOY.
BUOY
WATER MUST BE PRESSURIZED
wave w/ buoy
ME to PnE w/ buoy piston
air w/ turbine
AIR LIQUID
shoot w/ funnel
fluid w/ turbine
TURBINE
LIQUID (WATER, RAIN, URINE?) GOES THROUGH TO POWER TURBINE. COLLECTION BUCKET AT BOTTOM TO RE USE LIQUID FOR FUTURE POWER GENERATION
HARNESS AIR LEAVING SHOE WHEN STEPS TAKEN. HAVE FLAPPER OF SOME SORT TO RESTRICT AIR OUT/IN IN CERTAIN PLACES
AIR
MUST HAVE SMALL GEN DEVICE
foot/shoe
pressure w/ air cushion
air w/ flapper valve (supply)
fluid w/ bucket (collect)

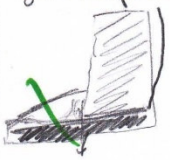
B2-PH1-B


GEAR OR
TURBINE - Turbine
POWERED
BY SAME
FROM
HOUSEHOLD
Elec
(convert)

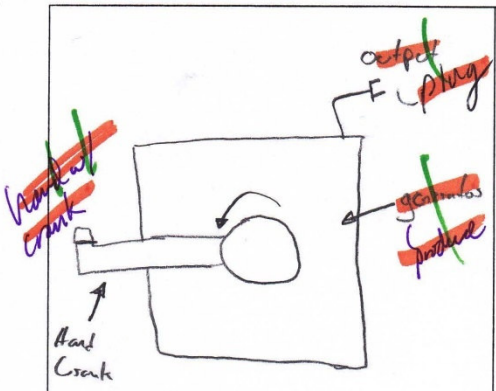
BS-PH-A

shoes that ~~operate~~ as pumping devices. Every step taken by the person would compress air. The compress air can be stored in a special backpack. "Buckets" of compressed air can be used to run generator at home.	Collect energy from vibration in human motion. Located on shoes.
~~Solar panel hat~~. Collect solar energy while working on rural areas. (Normally, under the sun) ~~Store~~ Photovoltaic, store energy in ~~battery~~.	~~If~~ there is an area where a lot of people walk through. Collect energy through displacements occuring by smart of people walking through.

B3-PHB-A

Mini tidal power generation Shoes used as pumping devices can replace the air blower constant pressure "backpack" to then use to run turbine and generate power. pressure out → tank transport air w/ turbine  foot w/ shoe pressure w/ boot pump	Air Blower device replaced by pumping device powered by weight of different people walking (or driving) through a monumented, or crowded area (or street) human w/ walkway
Vibration	

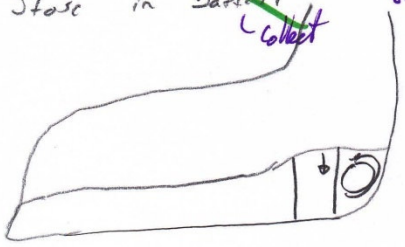
B4-PHI-A



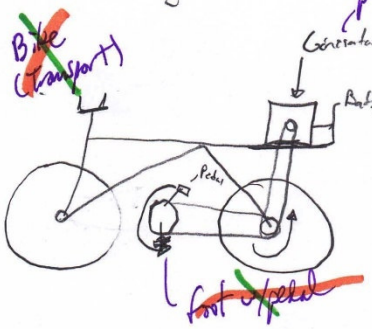
pressure w/ pump

Pump Shoes that turn produce small generator and Store in Battery collect

foot/shoe



Bike with crank generator

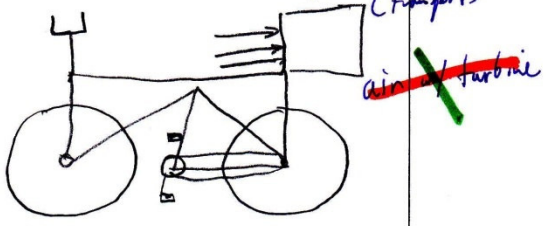


Person wearing Highly Static Clothing and then dis-charging into a Storage device

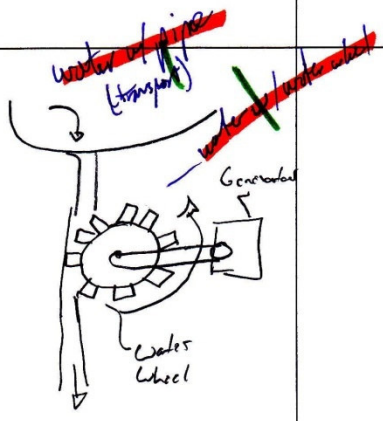
Static Elec w/ cloth (Produce)

B4-PHS-A

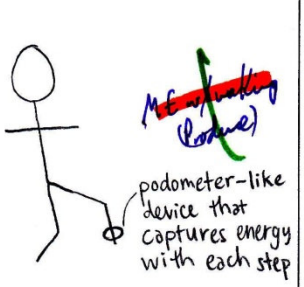
Wind turbine on back of Bike



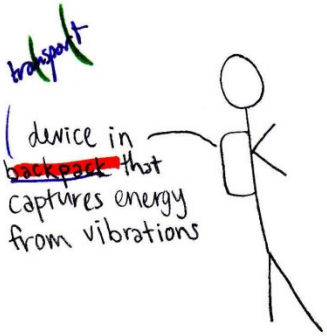
Suit that creates friction and transfers it into a storage unit for later use as deployment



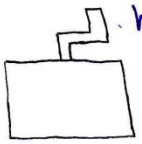
B5-PHI-A



podometer-like device that captures energy with each step



device in backpack that captures energy from vibrations



generator with a handcrank

device in soccer ball that captures energy while rolling

B5·PH3·A

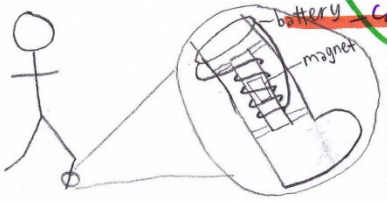
some kind of human-operated pump that compresses air for a small turbine

pressure w/ pump

air w/ turbine

water w/ waterwheel

pressure w/ pump



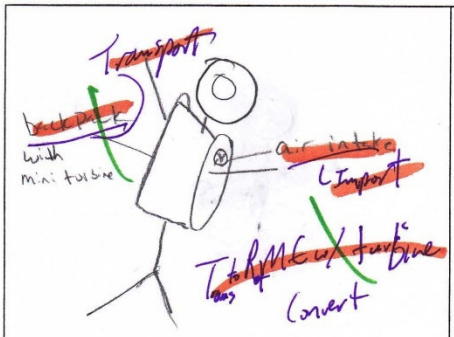
cell/cell

EE w/ magnet and coil

could be strapped to arm as well

strap w/ strap

B6-PHI-A

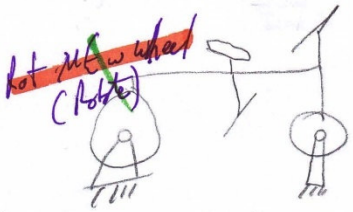


As person walks air flows into backpack which powers a mini turbine

Uses the up and down walking motion and change in PE to supply power to batteries integrated in brace that can be removed

Collect

Stationary bike power. Could be used as recreation as well for increase quality of life

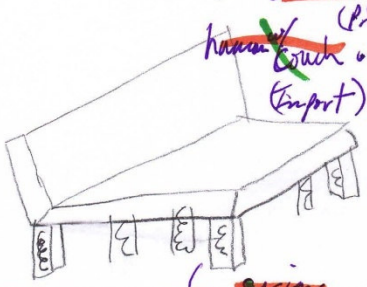


back wheel spins to create power

foot of pedals

couch that uses human weight to move magnets and force inductance charge

EE w/ mag/Induction (Produce)?



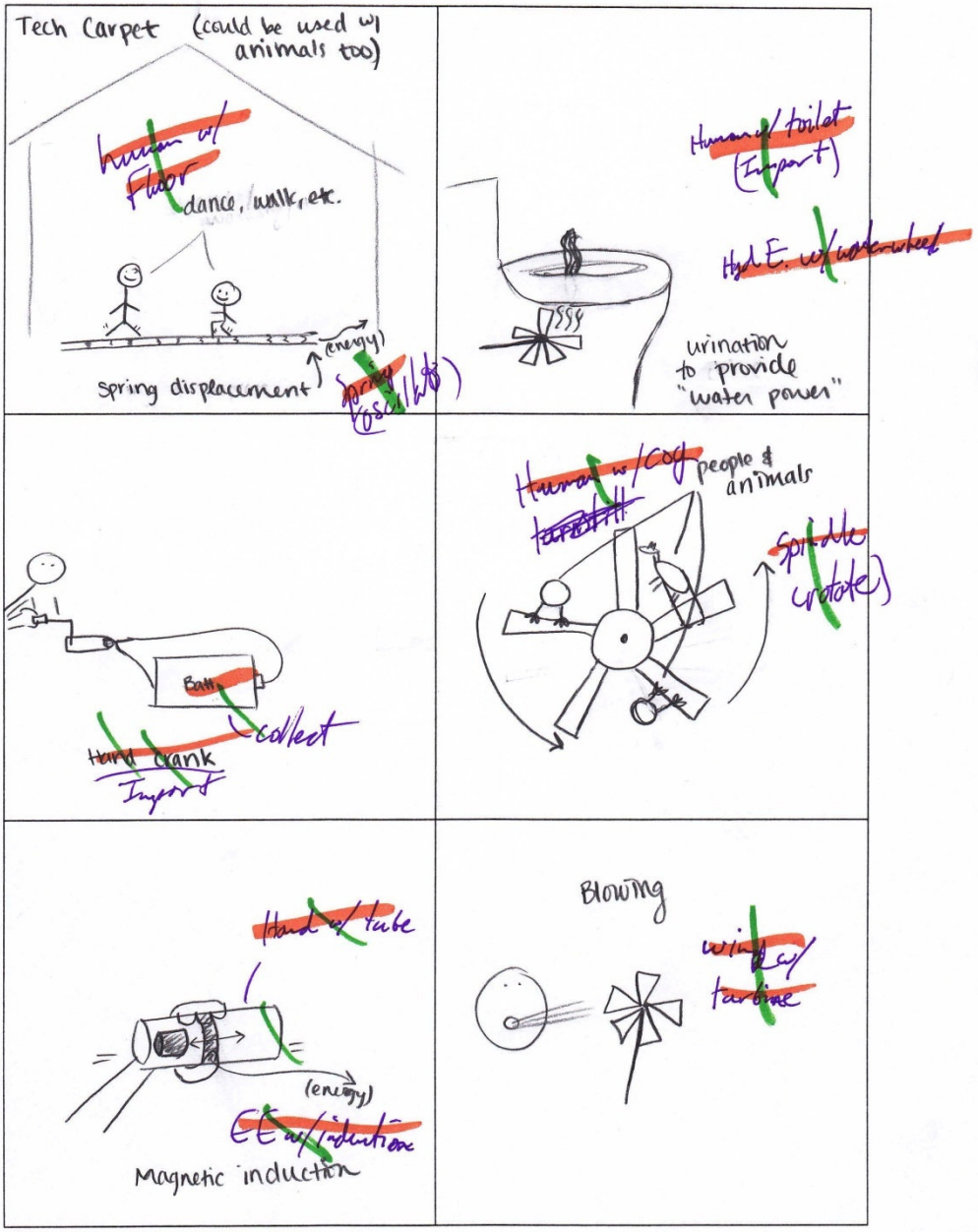
B6-PH3-A

Use human motion to tilt and rotate solar panels

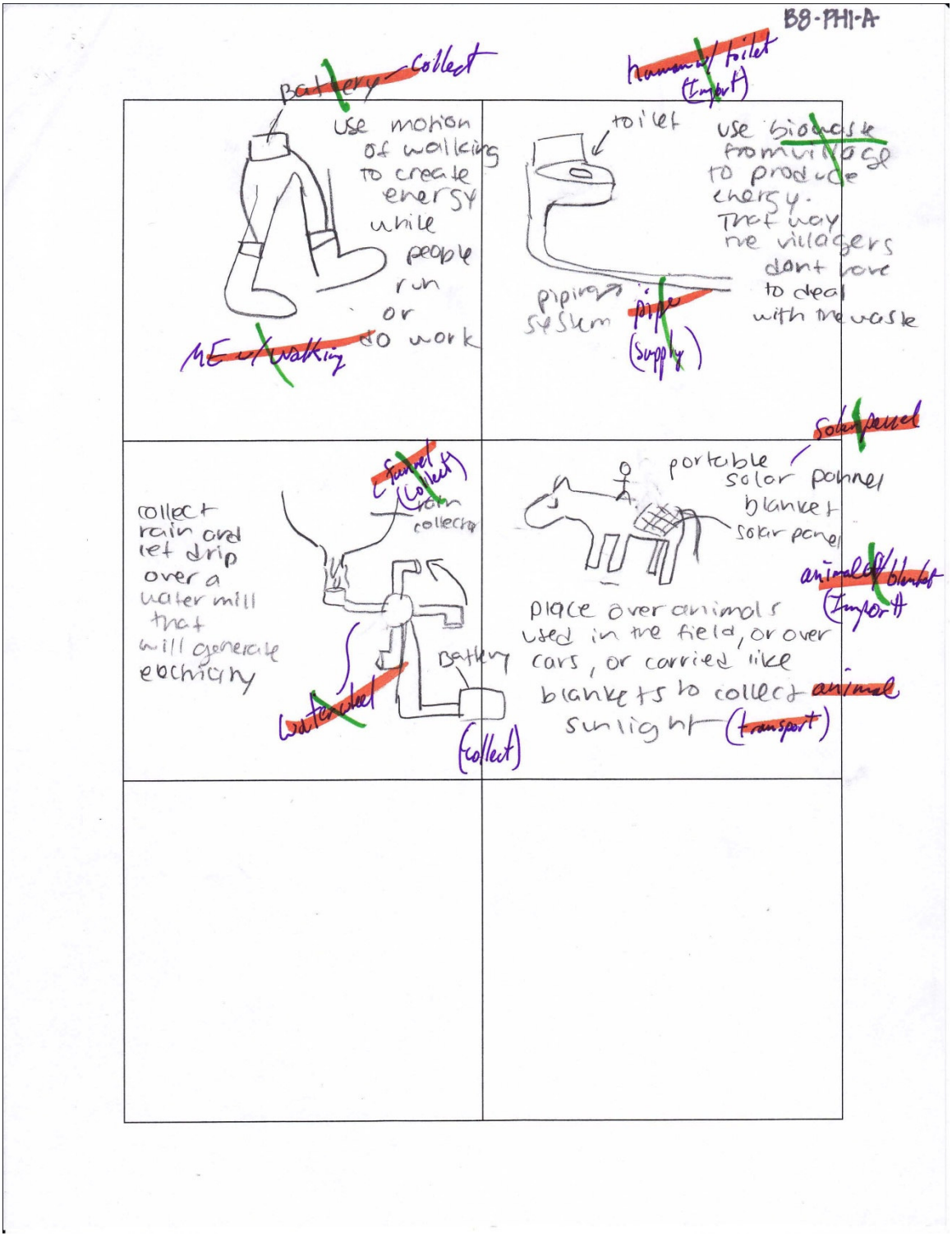
Attach a mini wave conversion unit to a human so when the swim it charges battery — collect

fluctuate with level of wave

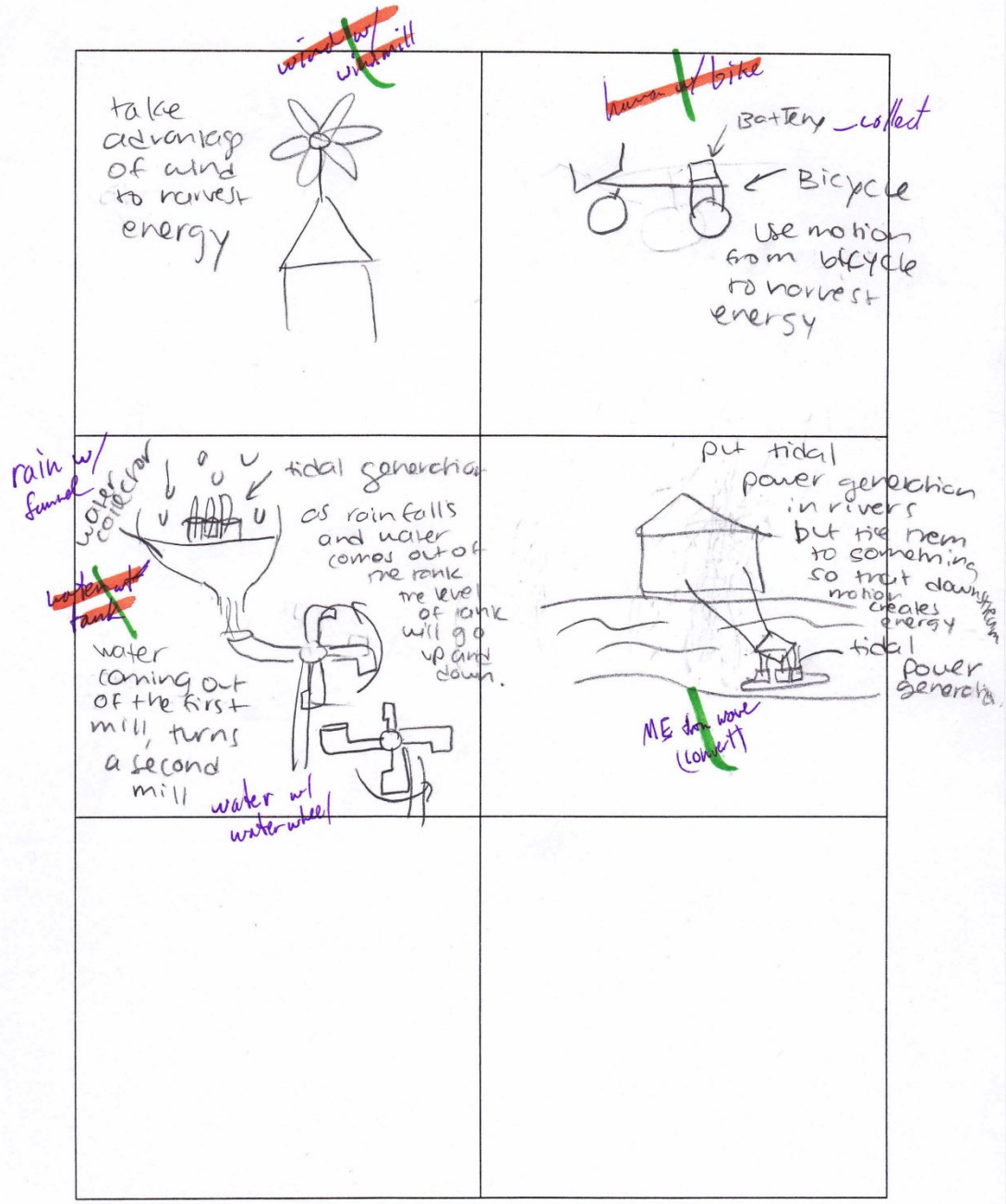
B7-PHI-A

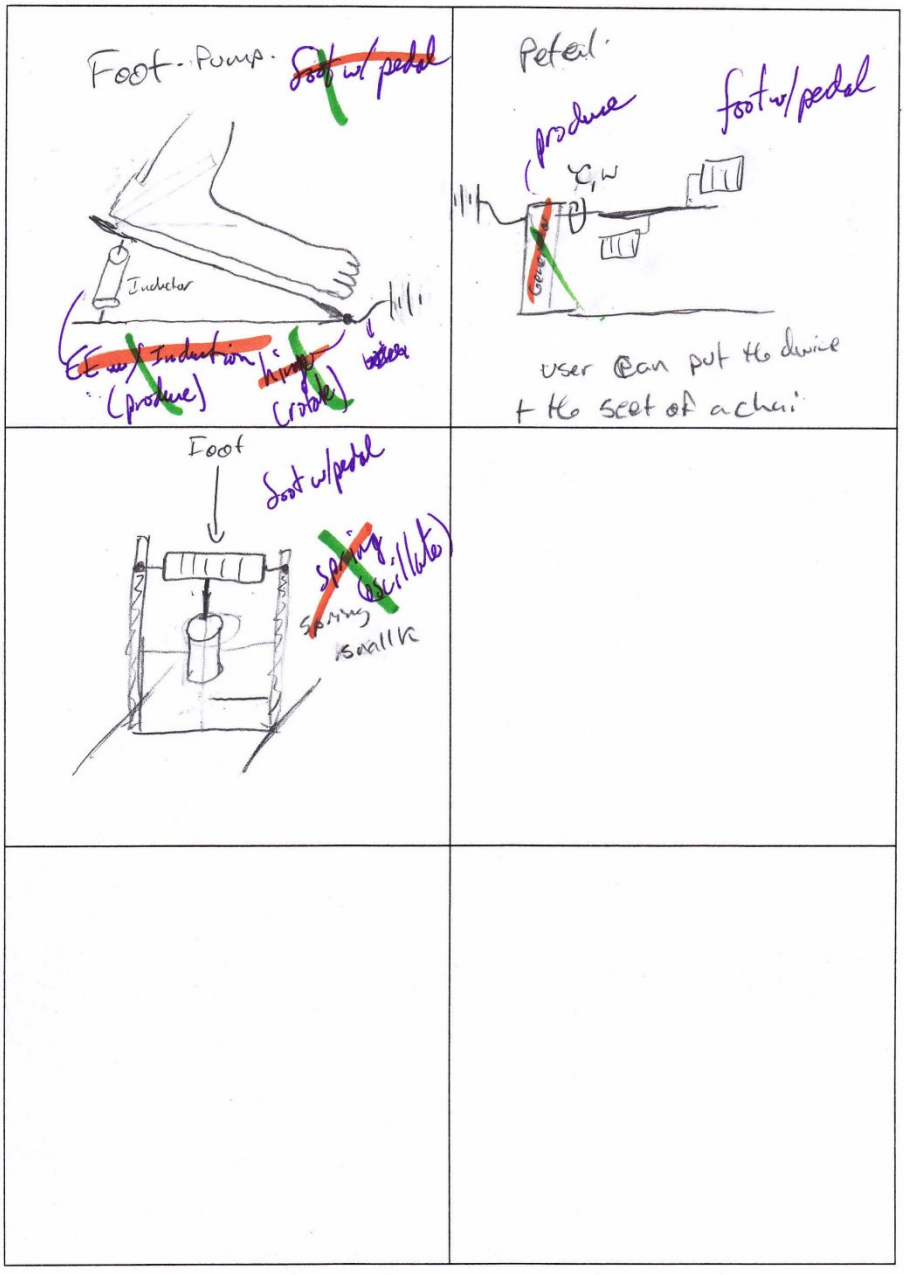

Tech Carpet (could be used w/ animals too)
dance, walk, etc.
spring displacement
(energy)
urination to provide "water power"
people & animals
Battery
collect
Hand crank
Important
Magnetic induction
Blowing
turbine

B8-PHI-A

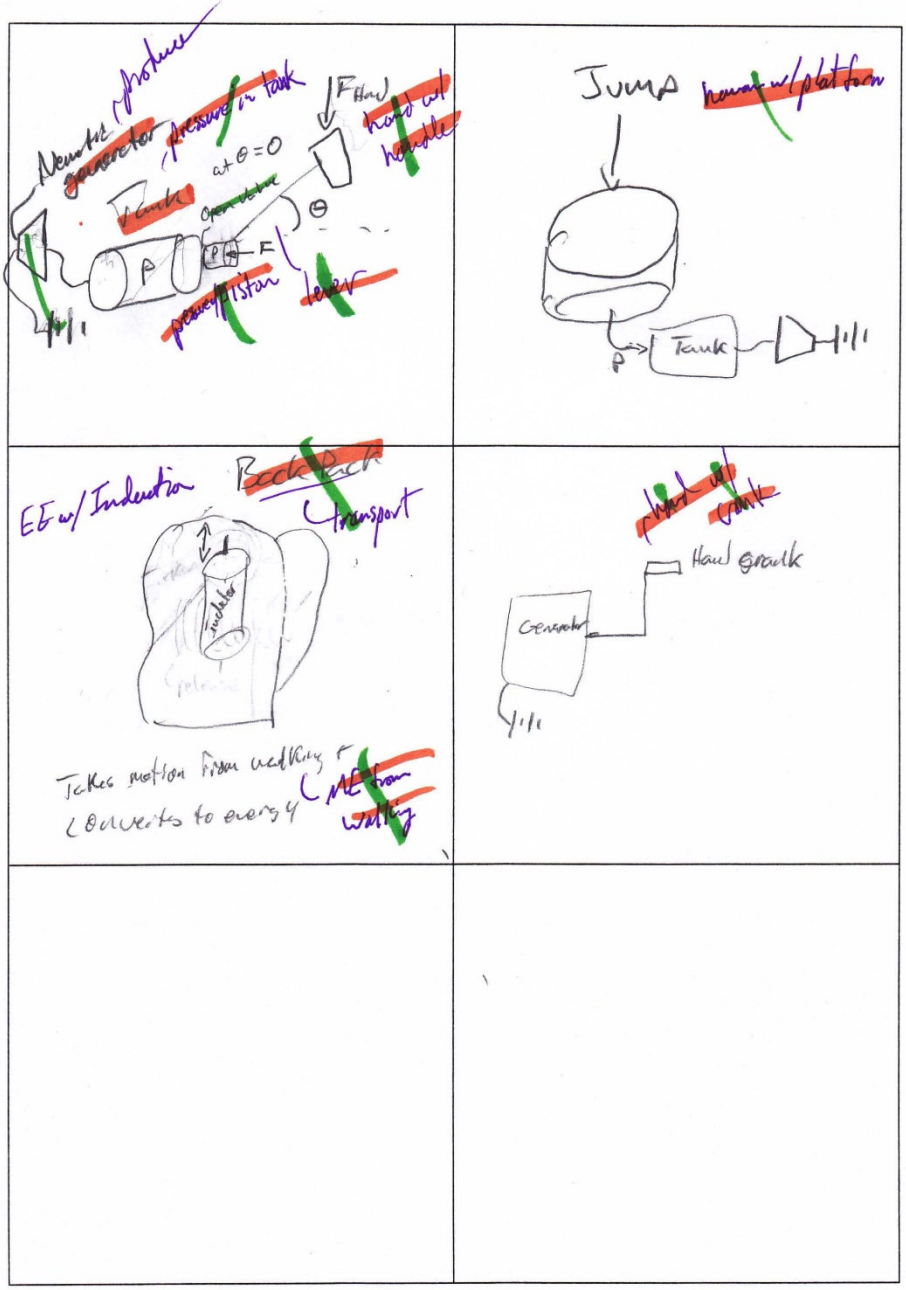

battery
collect
use motion of walking to create energy while people run or do work
ME walking
toilet
use biogas from human waste to produce energy. That way the villagers don't have to deal with the waste
piping system
pipe
(supply)
human toilet
(import)
collect rain and let drip over a water mill that will generate electricity
funnel
(collect)
rain collector
battery
water inlet
(collect)
portable solar panel blanket
solar panel
place over animals used in the field, or over cars, or carried like blankets to collect sunlight
(transport)
animal blanket
(import)
animal

BB-PHS-A

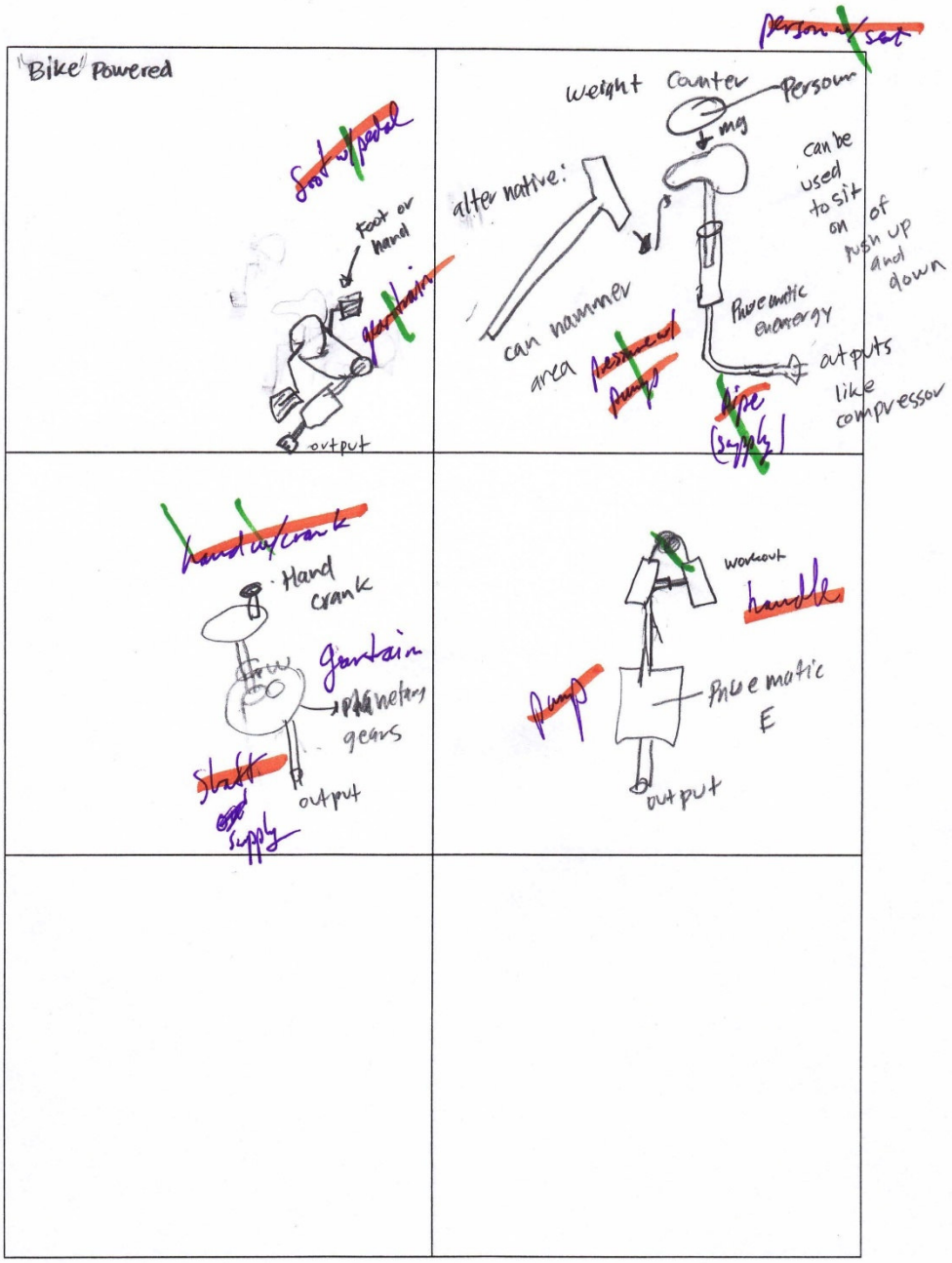

take advantage of wind to harvest energy
Battery collect
Bicycle
Use motion from bicycle to harvest energy
rain w/ funnel
tidal generation
as rain falls and water comes out of the tank the level of tank will go up and down.
water coming out of the first mill, turns a second mill
water w/ water wheel
put tidal power generation in rivers but tie them to something so that downstream motion creates energy
tidal power generation
ME from wave (convert)


Foot-Pump
foot w/pedal
Inductor
Induction
(produce)
(rotate)
Pedal
(produce
foot w/pedal
user can put the device
+ the seat of a chai
Foot
foot w/pedal
spring
oscillate)
small k

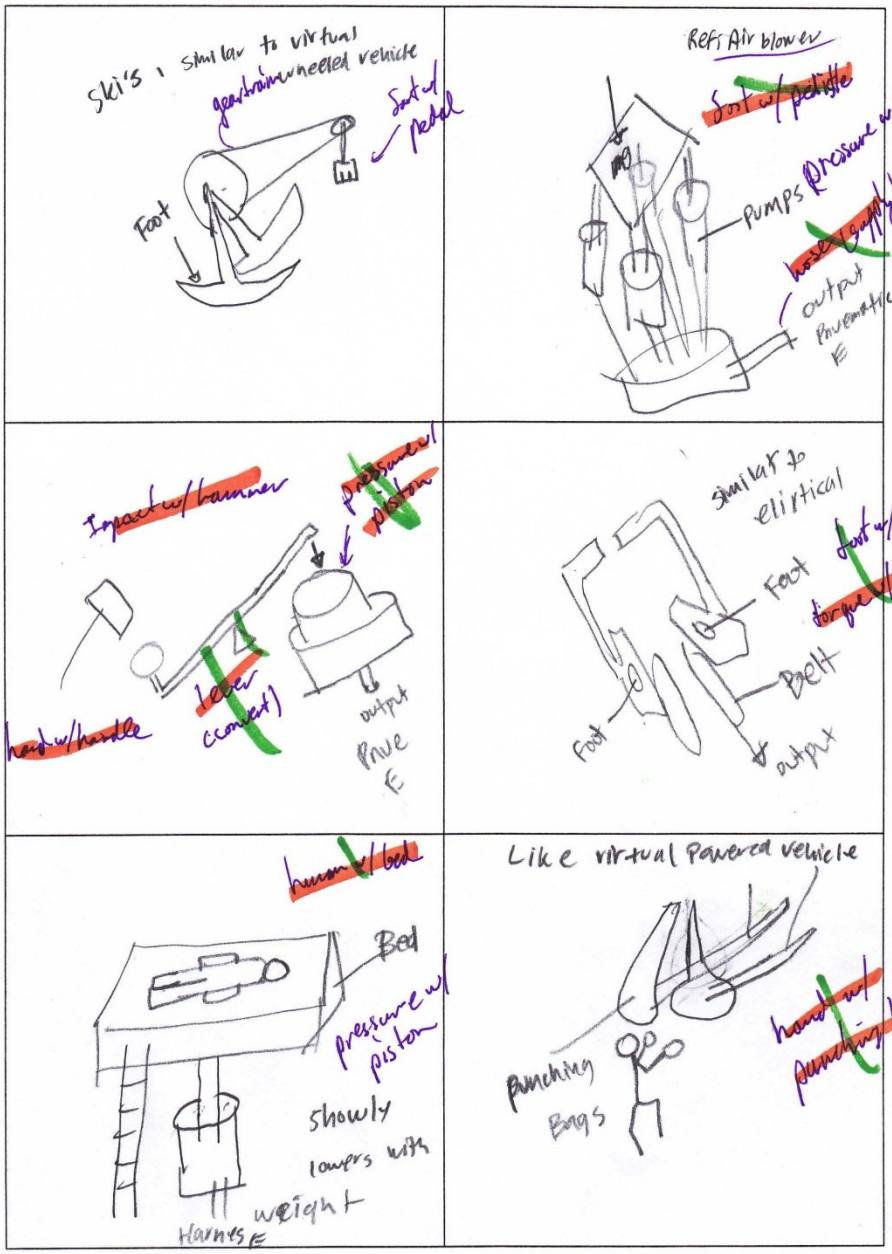
B9-PHB-A


Jump
Tank
P
EB w/ Induction
Rack
Pinion
Transport
Inductor
Takes motion from walking &
converts to energy
Haul gravity
Generator

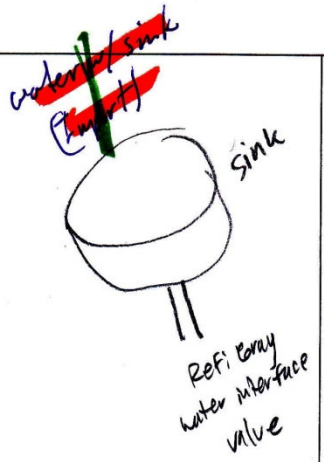
BIO-PHI-A


Bike Powered
Foot or hand
output
weight
Counter
Person
mg
alternative:
can be used to sit on of push up and down
can manner area
Pneumatic energy
outputs like compressor
Hand crank
gearbain
planetary gears
output
workout
handle
Pneumatic
E
output

BID-PH3-A


Skis, similar to virtual
gear train wheeled vehicle
Foot
Foot pedal
Refs Air blower
Foot w/ pedal
Pumps (pressure w/pump
output
Pneumatic
Impact w/ hammer
pressured
piston
lever
output
Pneu
similar to
elliptical
Foot
Foot w/pedal
Belt
Foot
output
human w/ bed
Bed
pressure w/
piston
slowly
lowers with
weight
Harness
Like virtual Powered vehicle
punching
Bags
hand w/
punching bag

BIO-PHS-B



EC w/ Induction

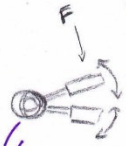
magnet can travel inside the coils by shaking the device, this creates a charge & stores it to internal battery.

(collect)

Bike powered generator can produce & fill up any other

(foot w/ pedal)

Foot w/ Step



Stepper that generates electricity

(Engage (rotate))

(Export) MG w/ ball (Import)

- Soccer ball
- kick it around & it will charge up. can plug other devices into it.

A bike with a small generator on it.

(produce)

bike (transport)

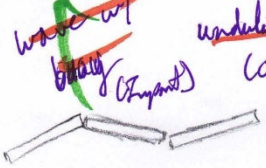
- device that does the opposite of the Peltier effect.
- basically heat generates electricity.

E from heat engine (thermocouple) (produce)

B11-PH1-B

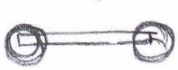
- a better rechargeable battery. Battery technology Cobalt has not advanced as quick as energy preservation.	

B11-PH3-A



Oscillating buoys that generate electricity. ~~connected~~ Waves ~~hooked up to a ~~ generate the oscillations.

Paddle boat that generates electricity. Or outfit an existing boat with a human powered ~~paddle~~ generator + paddle.

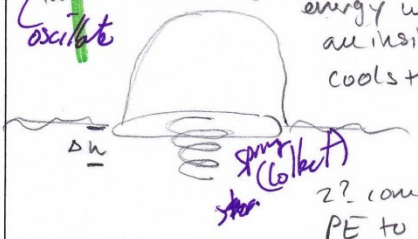


A cart when you pull it, the wheels turn the small ~~generators~~ on it.

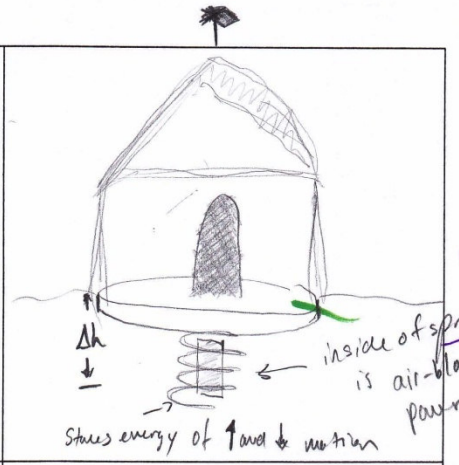
-a potentiometer hooked up to a rain collection system. The more rain, the more ~~charge~~ electricity being charged.

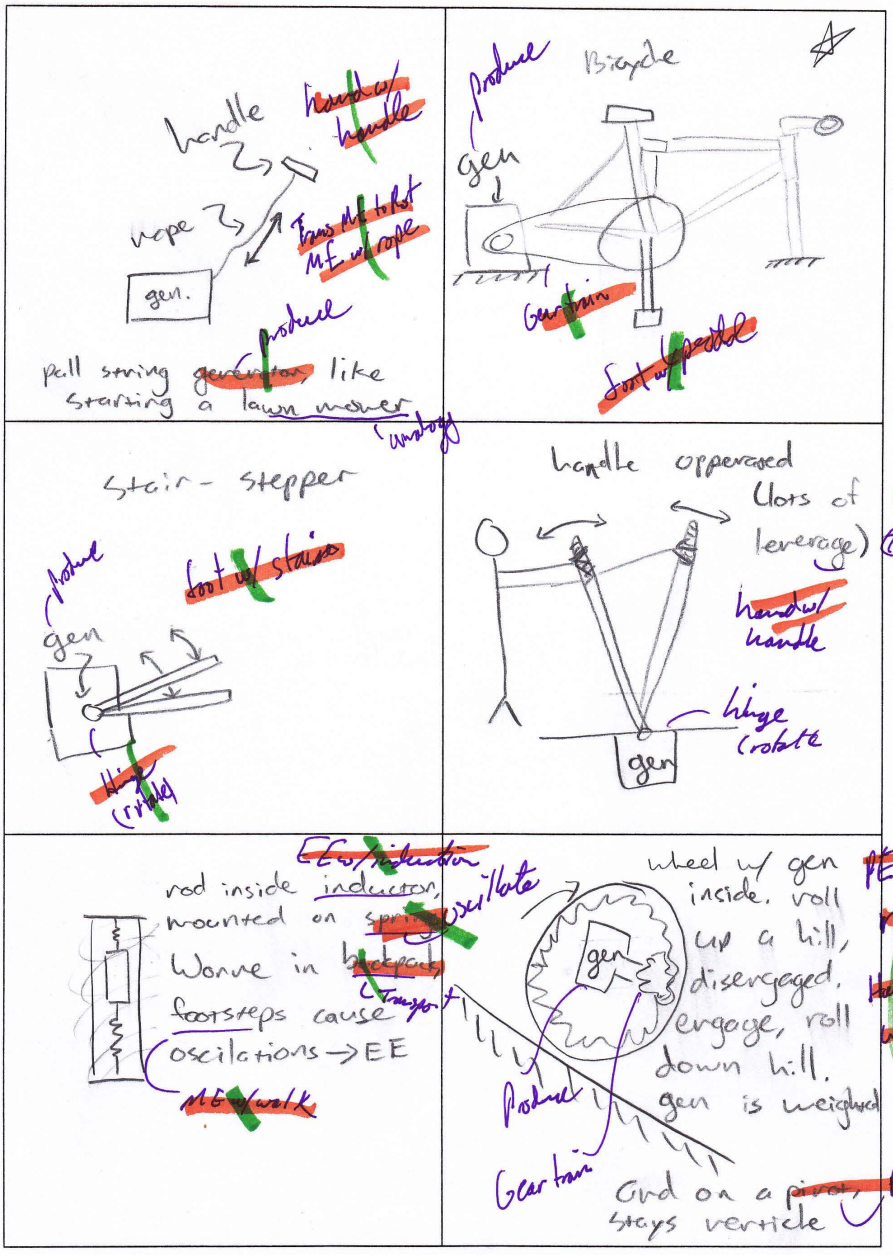
-photovoltaic spray paint

BI2-PHI-A

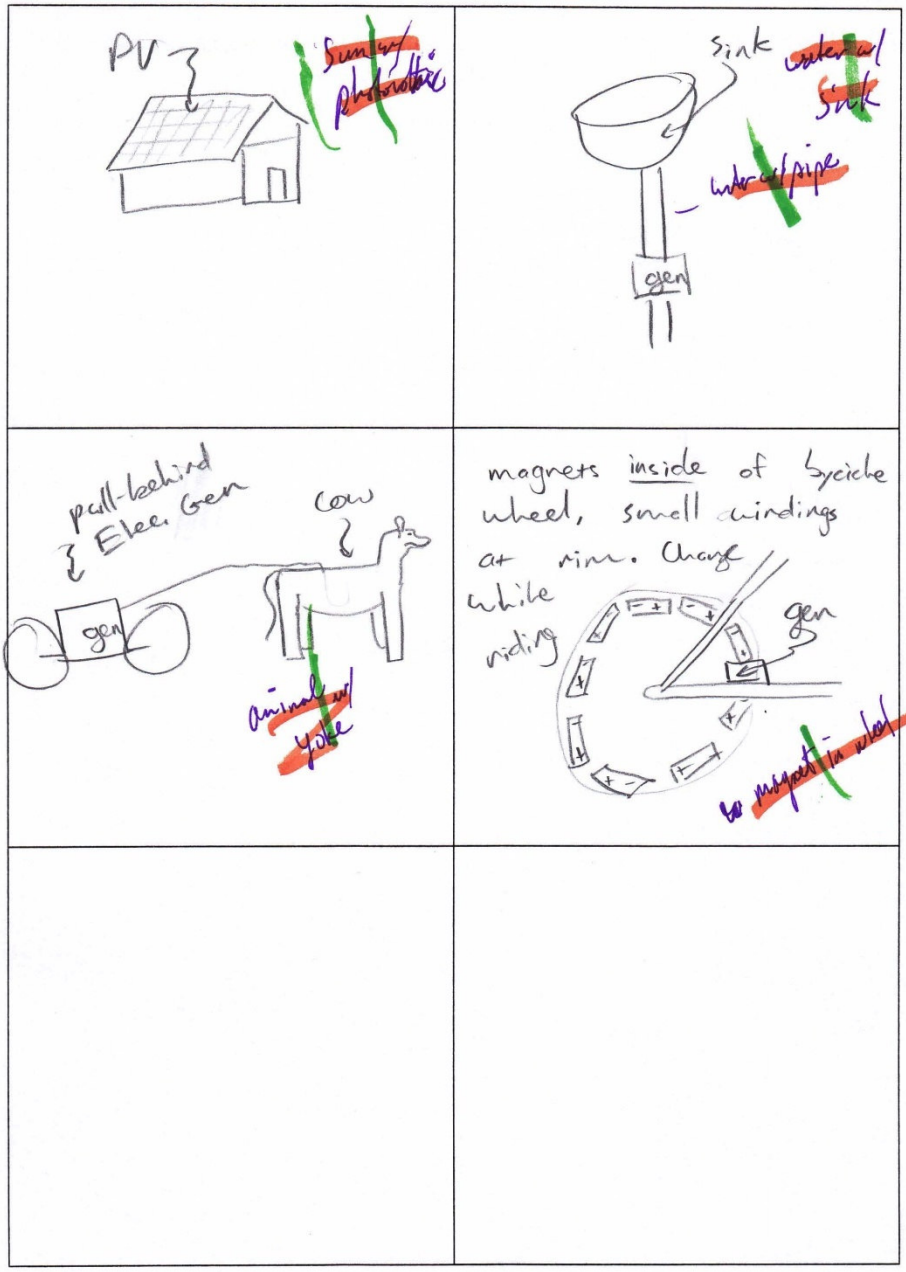
- gears ~~are~~ (produce) attached to their bicycles to store energy, even stationary bikes bike (transport)	- have the floor of their homes attached to giant ~~spring~~ that when they enter ~~lowers~~ the springs, stores potential energy while they are inside, and cools then is lowered. oscillate  Δh spring (collect) store 2? count PE to EE?
- give all people pedometers that stores peoples movements as energy, like a wave ~~ME of walking~~	-

B12-PH3-A


Δh
inside of spring is air-blower tidal power
Stores energy of ↑ and ↓ motion


handle
rope
gen.
pull string like starting a lawn mower
Bicycle
gen
handle
lever
hinge
gen
stair-stepper
gen
foot
gen
rod inside
Worve in
footsteps cause
oscilations → EE
wheel w/ gen inside. roll up a hill, disengaged. engage, roll down hill.
gen is weighted
Rotate

BIS-PHS-A


PV
Sun w/ photovoltaic
Sink
water w/ sink
water w/ pipe
gen
pull-behind Elec. Gen
gen
cow
animal w/ yoke
magnets inside of bycicle wheel, small windings at rim. Charge while riding
gen
to magnets in wheel

B-14

B14-PH1-A

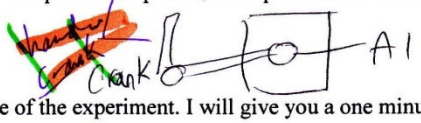
Designs Below ↓

Design Problem Statement

Design a device to collect energy from human motion for use in developing and impoverished rural communities in places like India and many African countries. Our goal is to build a low-cost, portable, easy to manufacture device targeted at individuals and small households to provide energy to be stored in a battery. The energy is intended to be used by small, low power electrical devices, such as a radio or a lighting device, hopefully leading to an increase in the quality of life of the communities by increasing productivity, connection to the outside world, etc.

Phase 1

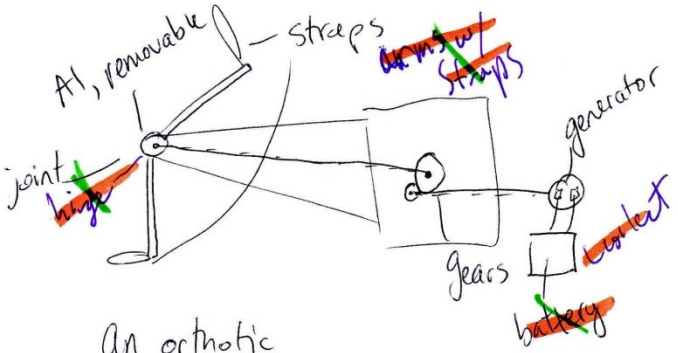
Generate as many solution concepts to the design problem as you can. Record all concepts, including novel and experimental ones. You may use words and/or sketches to describe your ideas. Please record each distinct solution concept in the separate boxes provided. Additional pages are available upon request.



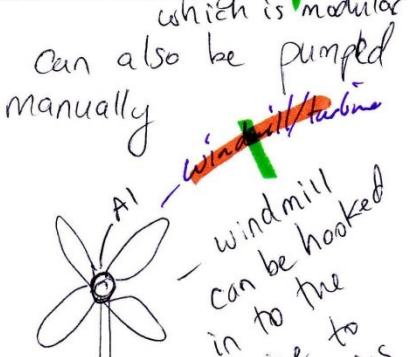
You have 10 minutes to complete this phase of the experiment. I will give you a one minute warning before your time is up.

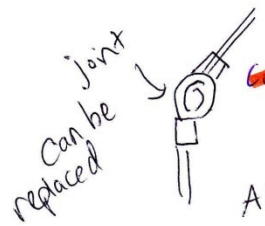
All of these use the same gear and generator which is modular

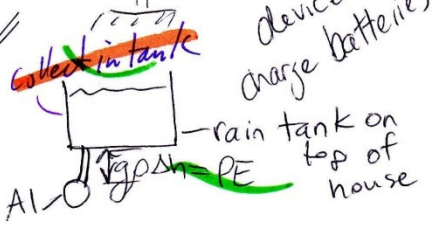
Please do not move on to the next section until instructed to do so.



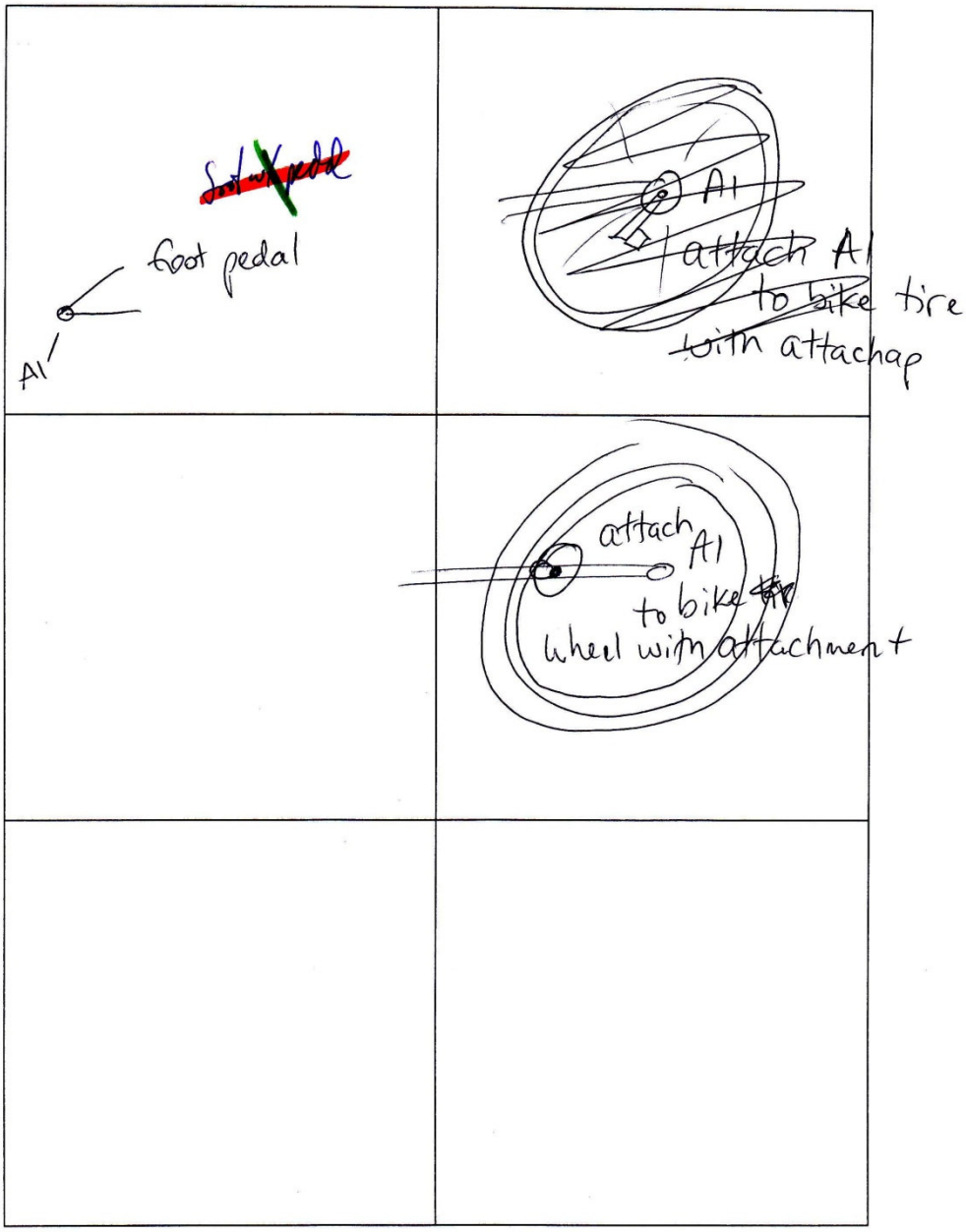
An orthotic device that connects to the leg of the user, charges batteries while you walk



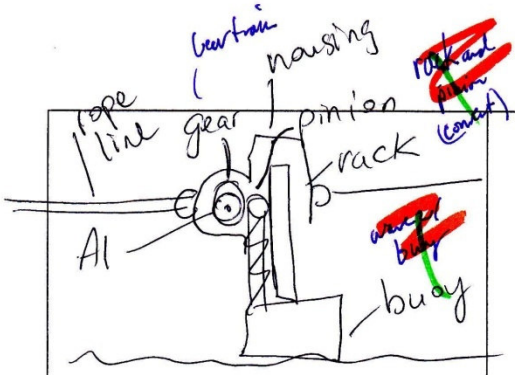




B14-PH1-B


foot pedal
A1
A1
attach A1
to bike tire
with attachap
attach
A1
to bike
wheel with attachment

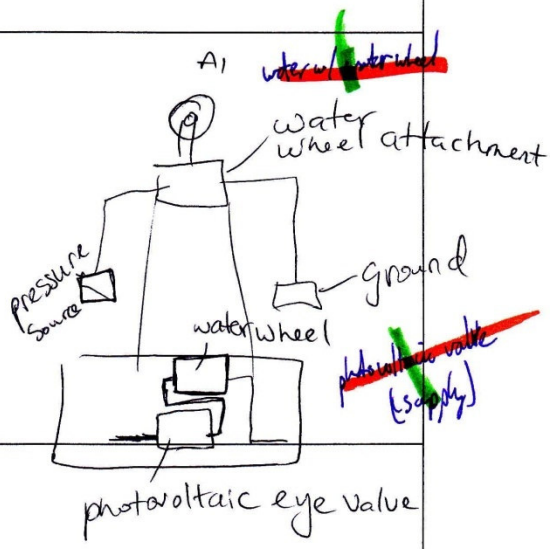
B14-P43-A

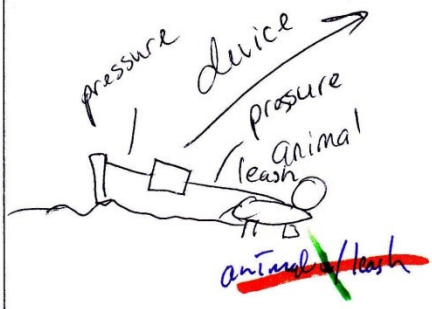


Charges via tides

Use A1 on a car wheel rather than a bike

Use A1 on a Cart wheel

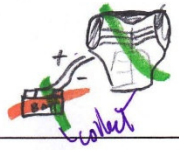




The device operates the same in both directions

Attach device from page 1 to cows, their legs that is 	

create a device that you wear at night like a shirt that uses piezo electric crystals to create electricity from when your chest compresses/ expands when you sleep. It'll be connected directly to your batt.



A door mat at each door way or a rug that uses the pressure created when people walk on it to generate electricity

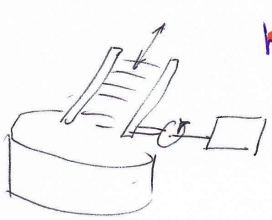
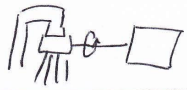
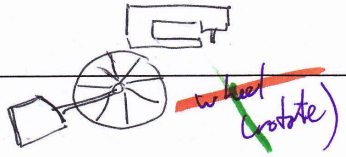
Small wearable devices you can attach to your wrist(s)/ ankle(s) that use your daily motions from performing activities to generate power, similar to watches that don't use batteries... except the power generated is saved in separate battery recipticles that when they become charged can be used similar to Double A or triple A batteries.

Attach electricity generating devices onto bicycle frames. when in operation during normal, daily activities it will charge removable batteries that will later be used to operate a small electrical device... This assumes that in impoverished countries they use bicycles to get around

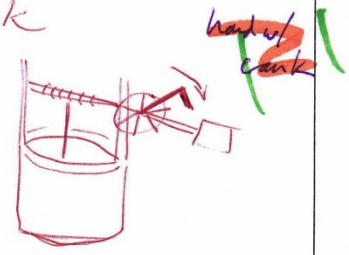
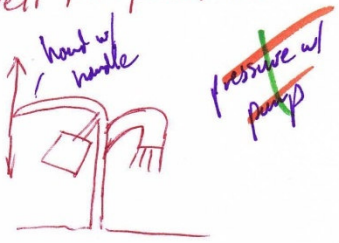
electricity generating device

BIS-PH3-A

Similar to the "air blower..." you can have a stationary bike that you pedal in order to wind an elastic element tightly. You would have a modulator on the shaft, such that when the elastic element wants to unwind it will do so in a constant velocity-manner. This will be coupled to a turbine to an electricity generator device that will produce a constant output.	Small photovoltaic that charge removable battery packs that are to be worn during daily activities.

Washboard collector - uses the motion of cleaning clothings on a wash board 	Door Collector - Uses the motion of opening + closing door to collect energy
Water flow - have a device connected on the out-let of a water pipe which transmits the flow to rotational motion which is the energy source 	For agricultural area use energy already being used to sow feilds to build up energy
Textile devices - Mechanical sewing machines - Mechanical looms → use rotational motion a device could also use this motion to collect energy 	

BIO-PHS-A

~~Rocking chair~~ movement to collect energy (oscillate)	Device on a well crank 
~~Well~~ pump device  → could combine with water outlet idea	Run off collector → using a device similar to patent #3 where the pressure build up of the run off is converted to electrical energy

B17 · PH · A ·

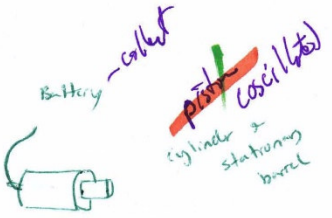
Use a crank to generate power, using hands or legs.

Device that captures heat generated from individual. Thermocouples embedded in a shirt

Piezoelectrics on areas heavily walked through.

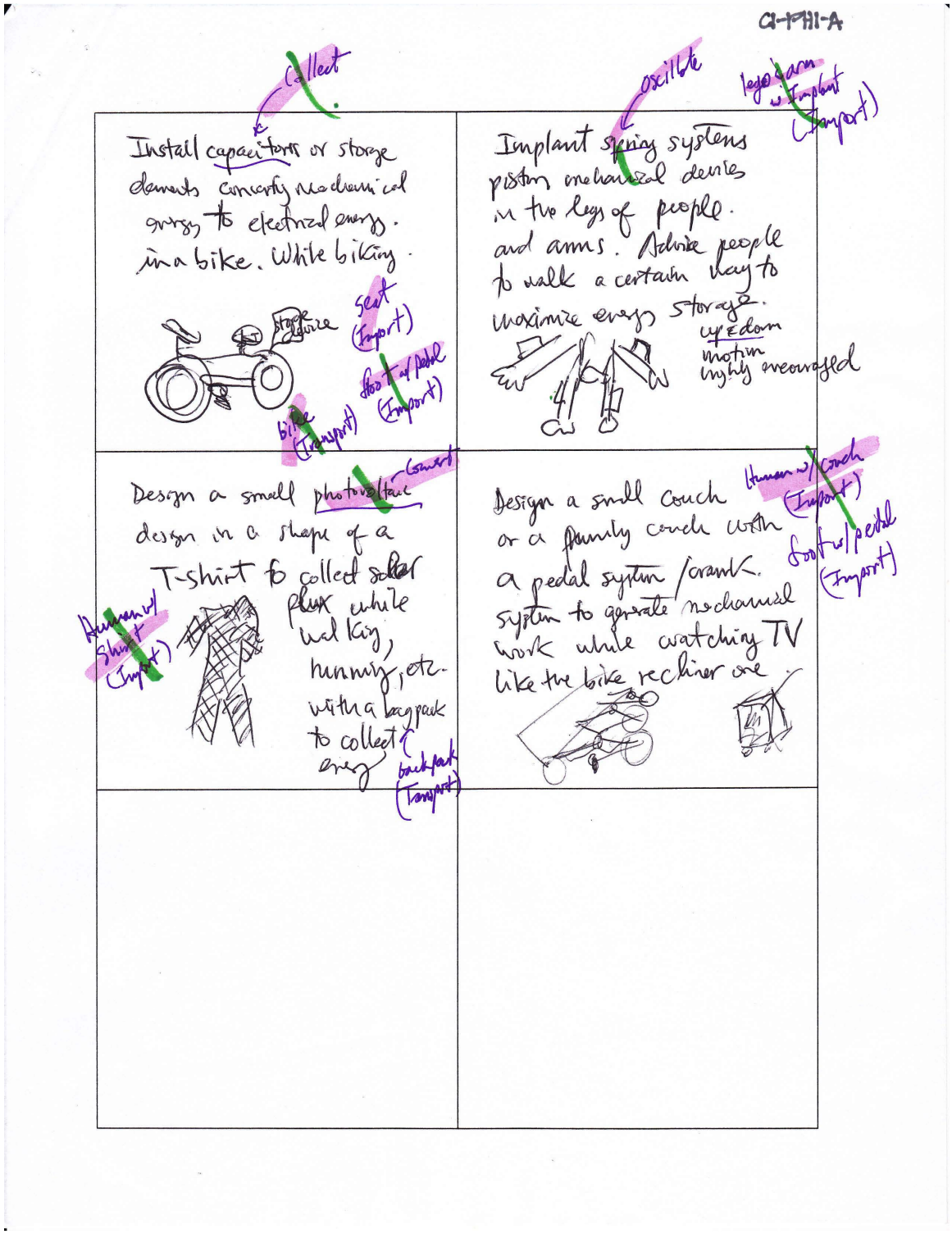
B17-PH3-A

Human Waste Reservoir
Power generated by turbine using waste's PE.

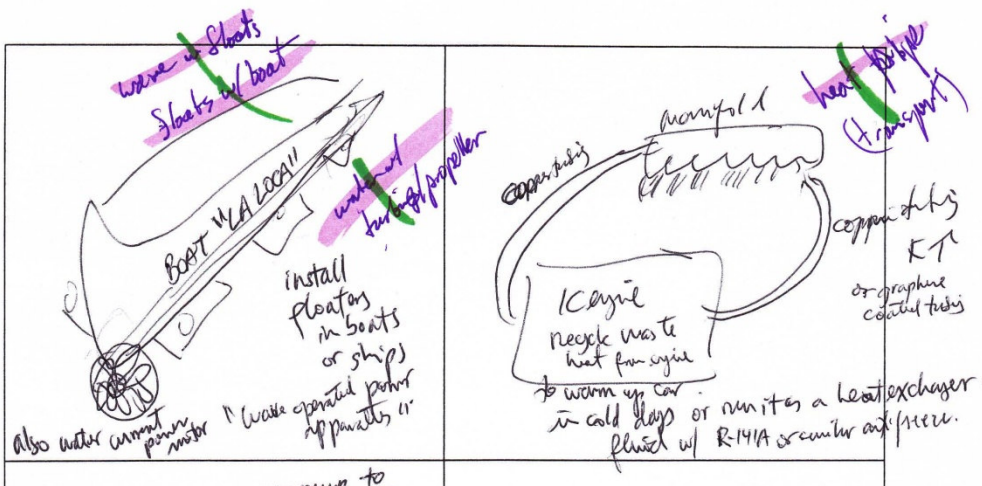


Use the "Motion of the Ocean"
Dudes eject their "jollies"!

GROUP C


C-PHI-A
collect
Install capacitors or storage devices converts mechanical energy to electrical energy. in a bike. While biking.
seat (Import)
bike (Transport)
foot w/ pedal (Import)
oscillate
leg/arm w/ implant (Import)
Implant spring systems piston mechanical device in the legs of people and arms. Advise people to walk a certain way to maximize energy storage.
up&down motion highly encouraged
Design a small photovoltaic (Generate)
design in a shape of a T-shirt to collect solar power while walking, running, etc. with a backpack to collect energy
Human w/ shirt (Import)
backpack (Transport)
Design a small couch or a family couch with a pedal system/crank system to generate mechanical work while watching TV like the bike recliner one.
Human w/ couch (Import)
foot w/ pedal (Import)

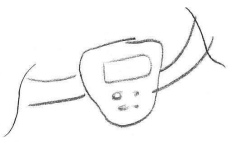
CI-PH3-A


wave & boats
boats & boat
BOAT W/A LOCAL
underwat turbine/propeller
install floaters in boats or ships
also water current power water
"water operated power apparatus"
copper tubing
manifold
heat pipe (transport)
copper tubing
KT
or graphene coated tubing
ICengine
recycle waste heat from engine
to warm up car in cold days or runs thru a heat exchanger. filled w/ R-141A or similar antifreeze.
pump to distribute heat around the house
or generate heat energy

C2-PH1-A

accelerometer (convert) human/w (Import)

Place accelerometer type devices on person that will store energy from up and down movements (i.e. walking, sitting down & getting up)



Place static charge rugs in households so that when walking electrical energy can be accumulated.

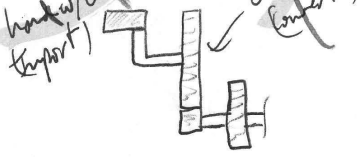
foot w/ mat (Import)

static charge (produce)

Have cranks which by turning can generate electricity.

human w/ crank (Import)

gear train (Convert)



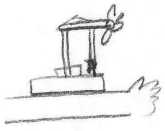
Develop children games that let children's energy to be harnessed.

(Ex. a ball that stores kinetic energy or rotation)

KE w/ ball (Collect)

Convert and store heat given off of human bodies to use for possible small sterling engines.

sterling engine (Convert) Collect?

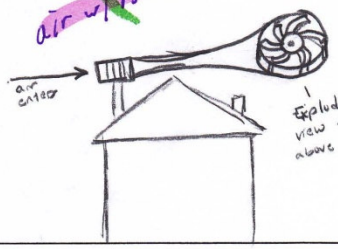


body heat w/ pad (Import)

C2-PH3-A

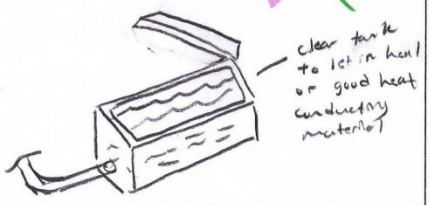
Have wind vane device on top of home to convert wind to electrical energy

air w/ rotor turbine



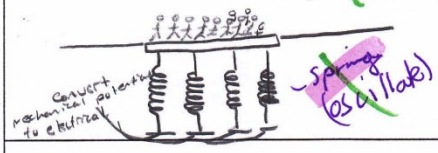
Use heat from sun (Africa & Asia usually hot) to heat up water during day to use as hot water at night.

hot water w/ sun



Since there are alot of people in ~~Asia~~ India can have a platform on springs in marketplaces that deflect and store energy from weight of people

human w/ platform



spring (oscillate)

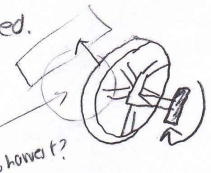
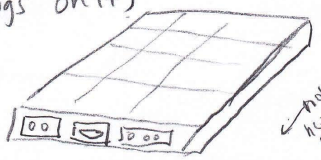
store the excess energy generated when riding a bike or other human powered transportation human w/ bike (pedal) seat (Import)	put a ~~hand~~ crank or ~~foot~~ pump on a chair to generate electricity to be stored. Import Pedal w/ pump (Produce)
take excess water collected from a well or river (from cooking, bathing, etc) and manually pour through water wheel and back into a bucket water w/ bucket (collect) ME w/ water wheel (Transform)	
use energy/ motion from children playing (swings, etc) and store for use into a battery human w/ seat (Import) swing (oscillate) collect	

C3-PHS-A

put a device on the bottom of fishing boats to sense wave energy & store it. waves w/ boat	use human movement (ex: walking) across the floor of a house, for instance, to cause a displacement and change in potential. human w/ floor ME from walking
after using a stove/cooktop/fire, put a device on top of the fire to convert water to steam until the heat runs out water to steam w/ fire	collect rain water from the roof and run it through a water wheel before it hits the ground to receive both rotational and potential energy. rain w/ roof (collect) water w/ water wheel
use the energy created by the motion in water caused by washing dishes, clothes, etc. in an enclosed area. waves from washing	use steam from cooking to extract energy from a small turbine ME w/ steam turbine

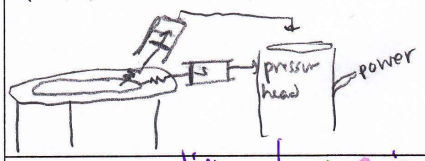
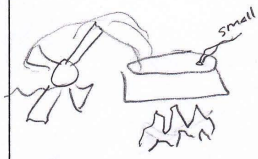
human energy from putting up clothes to dry outside in conjunction with weight and wind human clothes	

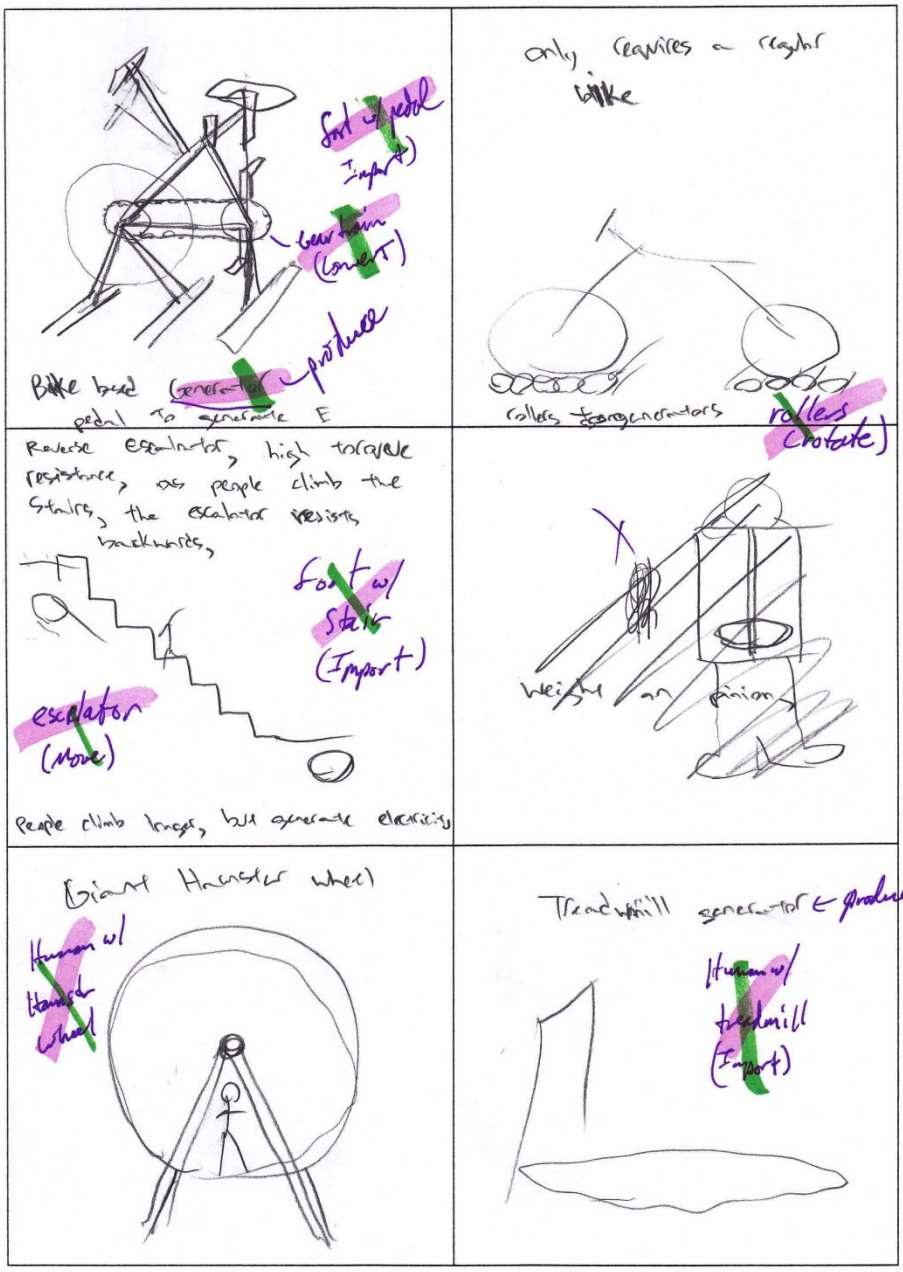
CA-PHI-A

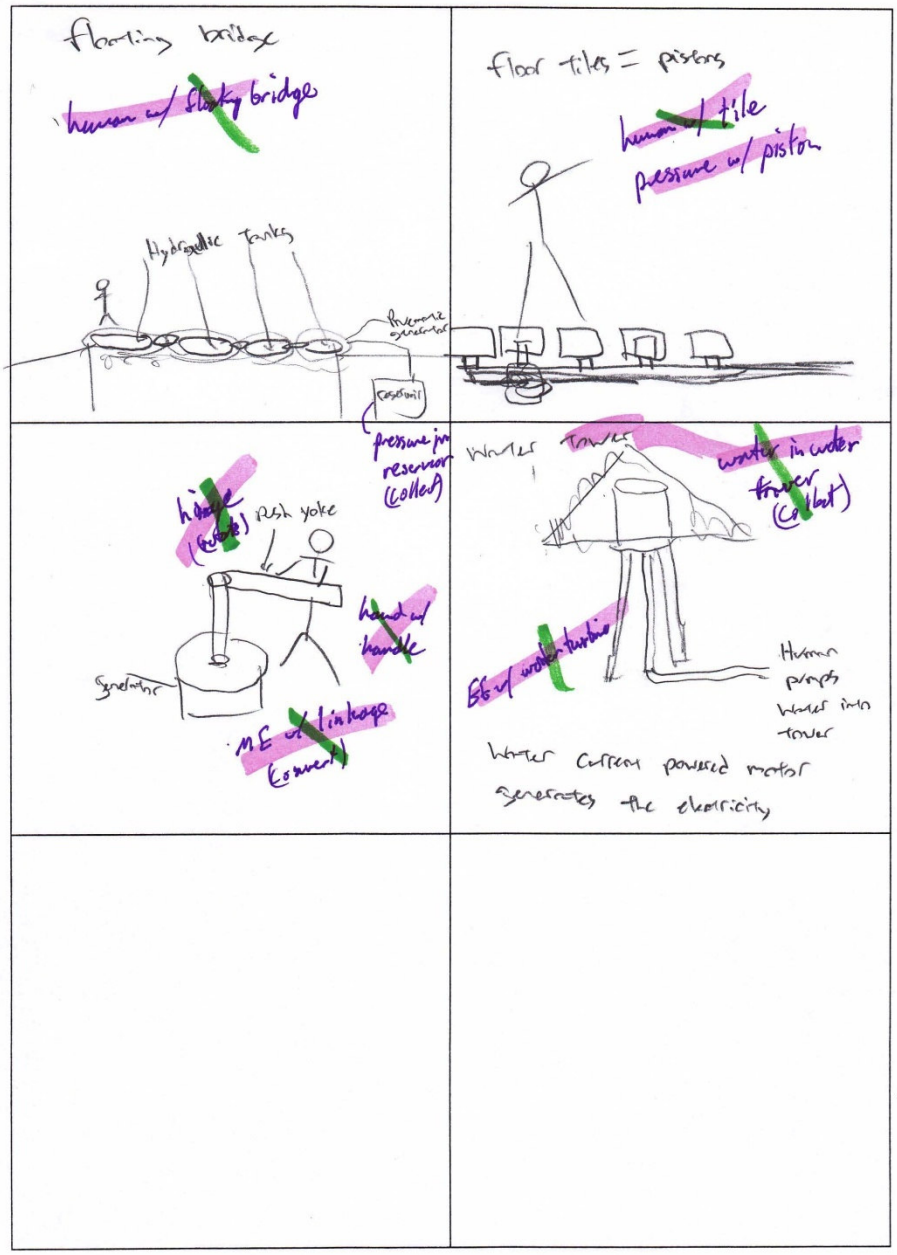
could use some sort of stationary bicycle that when used charges a battery. The faster you pedal the more charge. Also, could add resistance levels so the more resistance you use, also gives greater charge.	some sort of pulley charging system (like how you start a leaf blower/lawn mower) but that is small and could be pulled repeatedly and connected to some sort of power storage  when you let go it retracts so you can pull again
A crank/wind up device that could be spun (like a fishing reel) w/ some sort of resistance and the more it gets spun the more charge supplied.  not sure how to harvest?	small solar plate that harvests solar power during the day and can be used to charge devices at night. (plate has various plugs on it)  not human motion...
A small trampoline for children to play on, but the springs store the energy from jumping and it some how gets converted to usable electrical energy	Another childs toy that makes fun sounds when you shake it

C4-PHB-A

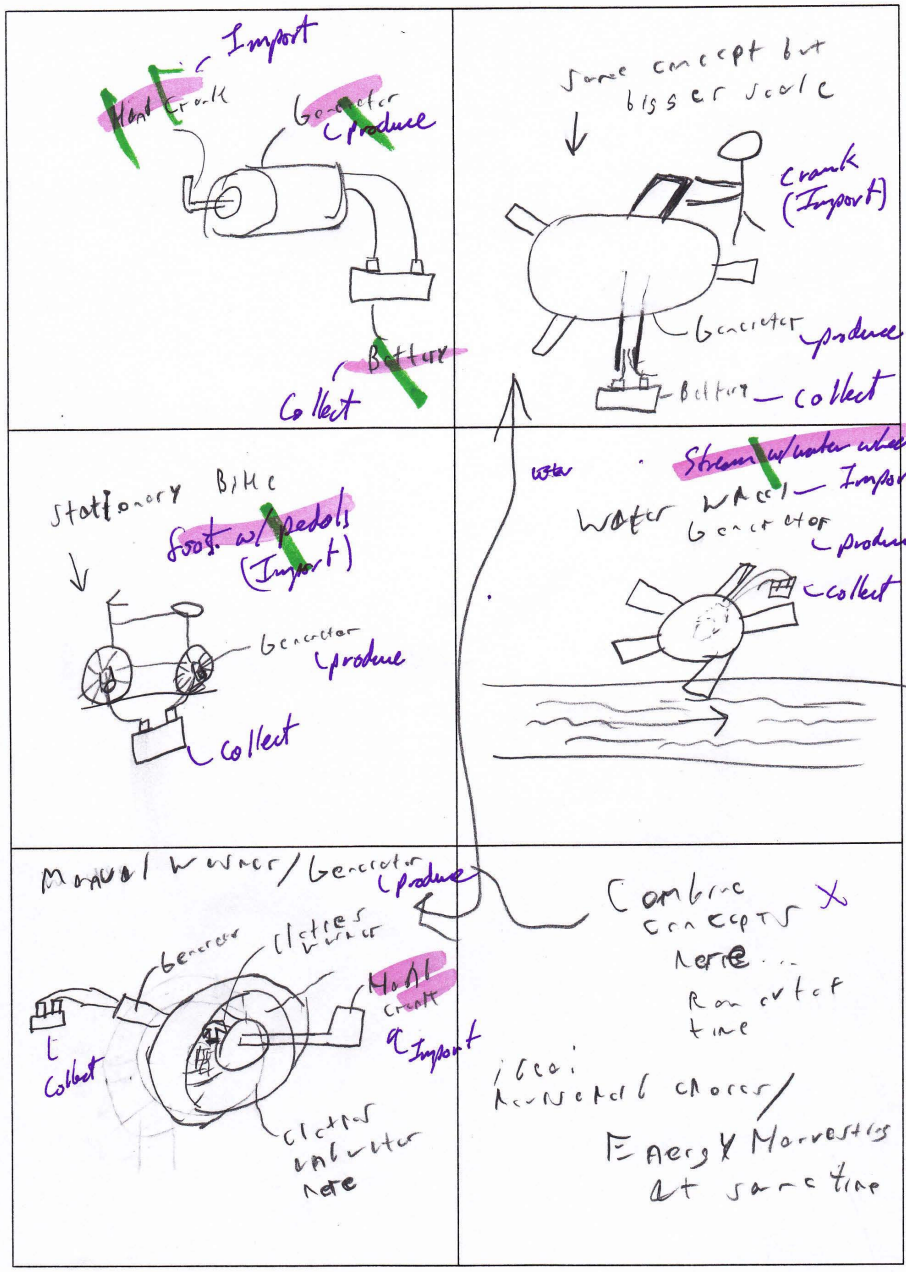
waves in pool

Combine patent 4 w/ stationary bicycle or crank. Can harvest water energy at the same time as human → don't have to have a water driven motor. For slow moving currents, may not get much energy on its own. water wheel w/ hand crank	Use patent one in portable baby pools. That way children can be playing, but also helping produce energy. Will have to be smaller, but b/c only wanting to ~~charge~~ power small household device, shouldn't pose too much of an issue.
Use patent 1 again, but only the pump idea. Combine w/ the trampolim idea from phase 1. the springs could be connected to pumps that arranged in "a pressure compounding relation" that fills a storage tank Coquillate pressure  human w/ trampoline compressed air w/ tank	Use part of the hybrid power generating assembly from patent 2 → the water injection engine. Can collect water w/ water mill then store in heated container then use water steam conversion  water to steam w/ engine water w/ water mill! (transport)


foot w/ pedal (Input)
gear train (convert)
produce
Bike based generator
pedal to generate E
only requires a regular bike
rollers + generators
rollers (rotate)
Reverse escalator, high traverse resistance, as people climb the stairs, the escalator resists backwards,
foot w/ stairs (Import)
escalator (move)
People climb longer, but generate electricity
weight on pinion
Giant Hamster wheel
Human w/ Hamster wheel
Treadmill generator ← produce
Human w/ treadmill (Import)

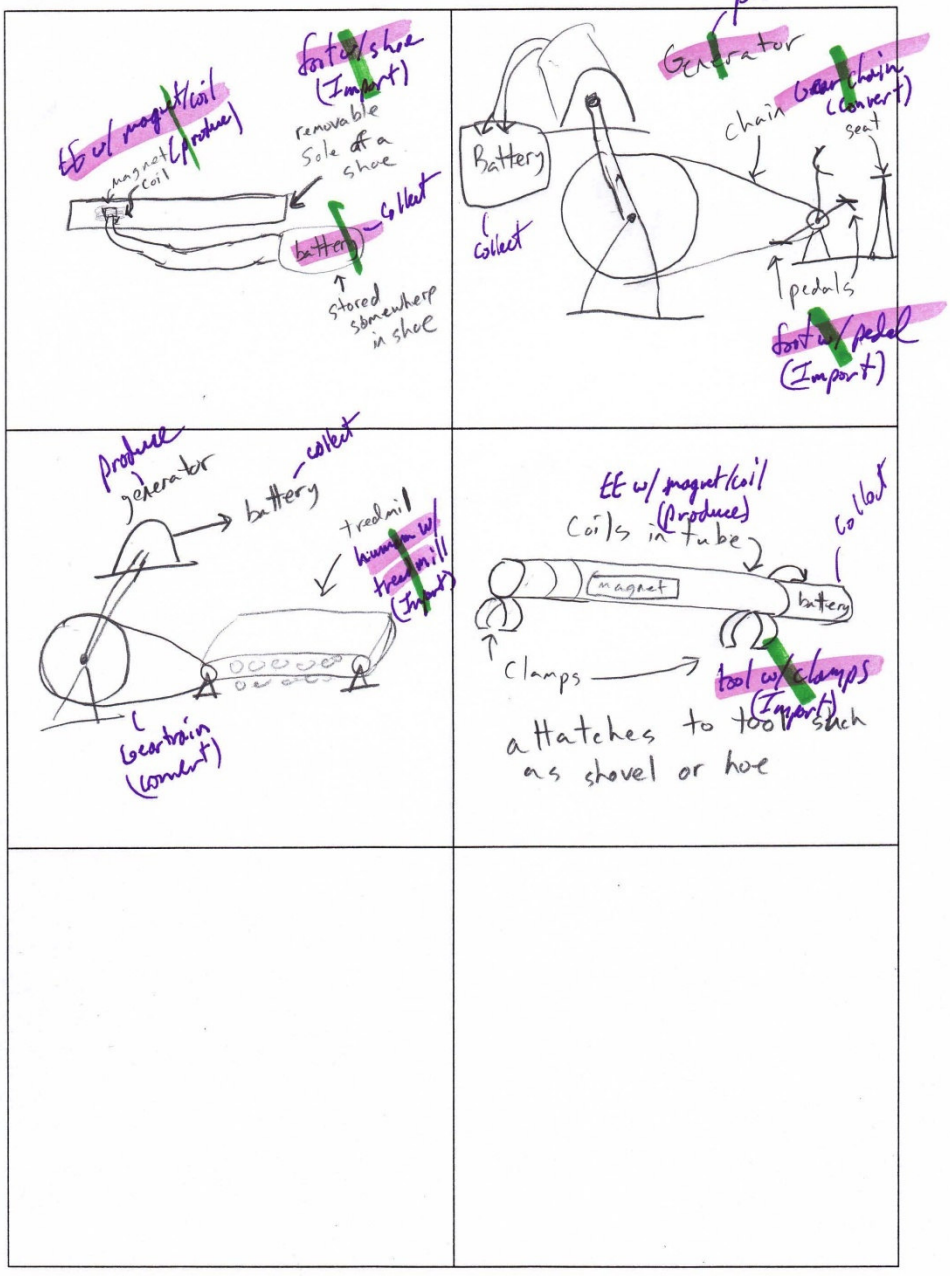

floating bridge
human w/ floaty bridge
Hydraulic Tanks
pressure in reservoir (cooled)
floor tiles = pistons
human / tile
pressure w/ piston
handle (wood)
Rush yoke
hand w/ handle
Generator
ME w/ linkage (sewed)
Water Tower
water in water tower (cold)
EE w/ water turbine
Human pumps water into tower
Water current powered motor generates the electricity

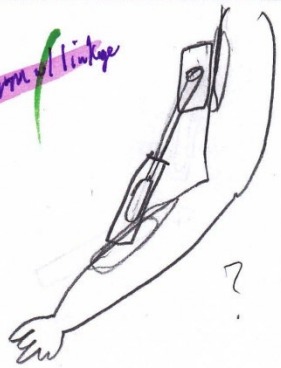
C6-PH1-A


Import
Heat Crank
Generator
Produce
Battery
Collect
Same concept but
bigger scale
Crank
(Import)
Generator
produce
Battery
collect
Stationary Bike
Foot w/ pedals
(Import)
Generator
Produce
Collect
water
Stream w/water wheel
Water wheel
Import
Generator
Produce
Collect
Manual Winder/Generator
Produce
Generator
Electric winder
Manual
Crank
Import
Collect
Electric
Calculator
Here
Combine
Concepts
Here...
Run out of
time
Idea:
Mechanical Clocks/
Energy Harvesting
at same time

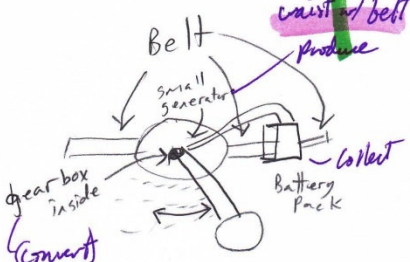
C6-PHS-A


Use valve concept from patent 1
button control or plow
water wheel generator
water w/ waterwheel
wheel from last patent
MS w/ walking
piezometer wave collector that stores energy
solar panels


EE w/ magnet/coil (produce)
magnet
coil
foot/shoe (Import)
removable sole of a shoe
battery
collect
stored somewhere in shoe
produce
Generator
chain
gear/chain (convert)
seat
Battery
collect
pedals
foot w/ pedal (Import)
produce
generator
battery
collect
treadmill
human w/ treadmill (Import)
gear/chain (convert)
EE w/ magnet/coil (Produce)
Coils in tube
collect
magnet
battery
Clamps
tool w/ clamps (Import)
attatches to tool such as shovel or hoe



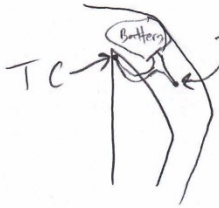
?



swinging weight

ME w/ weight (rotate)

Vest - that somehow catches body-heat



Temperature differance at Thermocouple creates tiny voltage

CB-PHI-A

- Similar to hand crank flashlights; have a device that is capable of storing mechanical energy into a battery for later usage. (good for exercise, too)

hand w/ crank (Input)

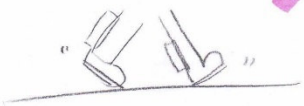
battery (Collect)

- Installing a device that can be attached to a bicycle. The device can collect wind energy from user's movement and the mechanical pedaling motion could provide energy.

bicycle (Transport)

foot pedal (Input)

Devices that can charge or store electrical energy from the human movement.
Eg: walking



Shoe (Input)

Walk (Output)

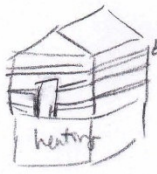
use flywheel and magnets (or polarity) to spin flywheel attached to shaft and run a generator that can convert that mech E to EE.

Collect

Produce

- Add wheels for portability
- Stronger materials for durability
- Of course there is solar, wind, etc...

wheels (Transport)

If resident has a pool, use buoyancy forces in pool. float w/ pool	Anaerobic digestion of organic waste. This will create compressed methane to use for heating. waste to methane w/ bacteria (current) gas w/ compression (collect)
• Implementation of the water powered motor in a region along a river would be energy harvesting. • could also be powered by water from cooking, showers, etc.	river w/ turbine
In addition to the heat generation for homes, designing homes, themselves, to be adaptable to weather conditions. Eg:  shutters. – open during summer to allow air flow through home. – close at night and allow heating. shutters w/ (current)	← (not ok) – glass is good too because it can heat up during the day and provide heat during the night. heat w/ glass (collect)

CG-PHI-A

footwear that stores energy from walking, like our military has

(walking, oscillate) (Import foot)

(Vehicle Import) path that gets relatively heavy traffic converts & stores vibrational energy (oscillate)

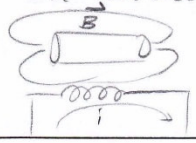
(collect)

glove or hand/arm attachment that captures, converts, & collects the energy from their motion during the day

(hand w/glove (Import))

perhaps exploit Faraday's law to accomplish this (run magnets across coils to generate a current

(EG w/magnet/coil (produce))



C9-PH3-A

Equip everyone w/ whirligigs in windy places. Make it charge a storage device

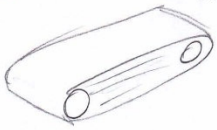
perhaps keep it indoors due to thieves.

Have a large storage device to home power converter

Inject NANO particles (magnetic) into people's blood streams and make them wear coils on their left armor neck to maximize the flux rate. Put that lazy blood to WORK! It'll charge their storage device (and hopefully give them more magnetic personalities)

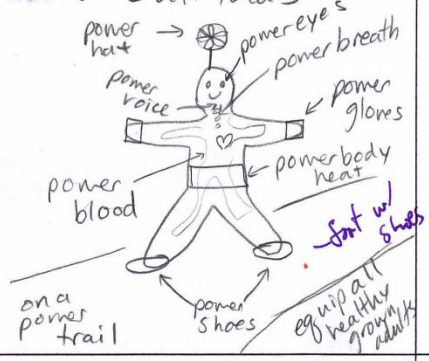
Wear elaborate arm/leg/body pieces with temperature sensitive metals (like thermistors) and convert body heat to electricity.

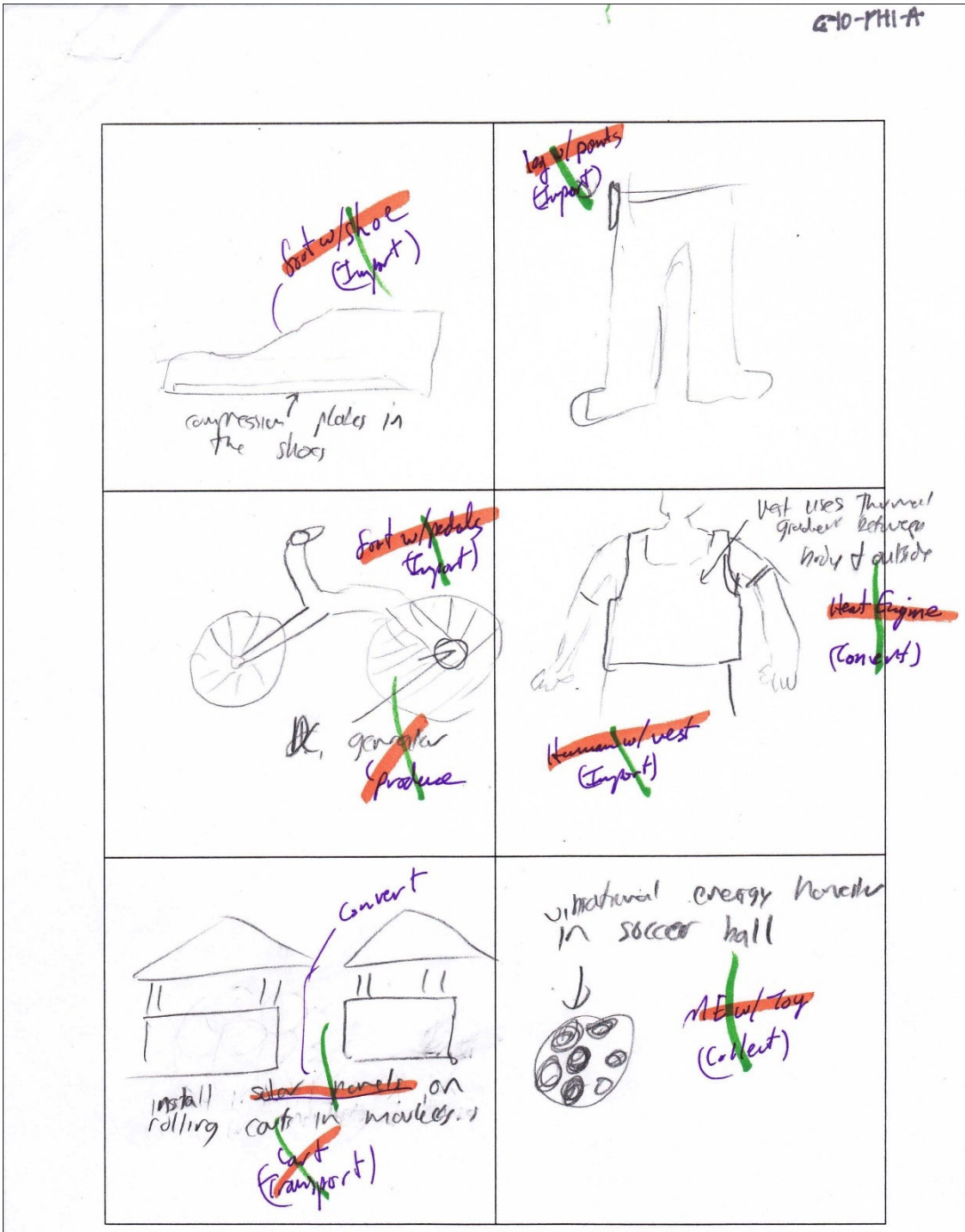
a simple treadmill device would suffic



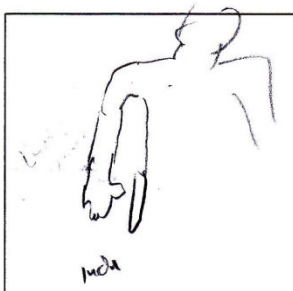
attach a device to people's vocal cords for energy collection.

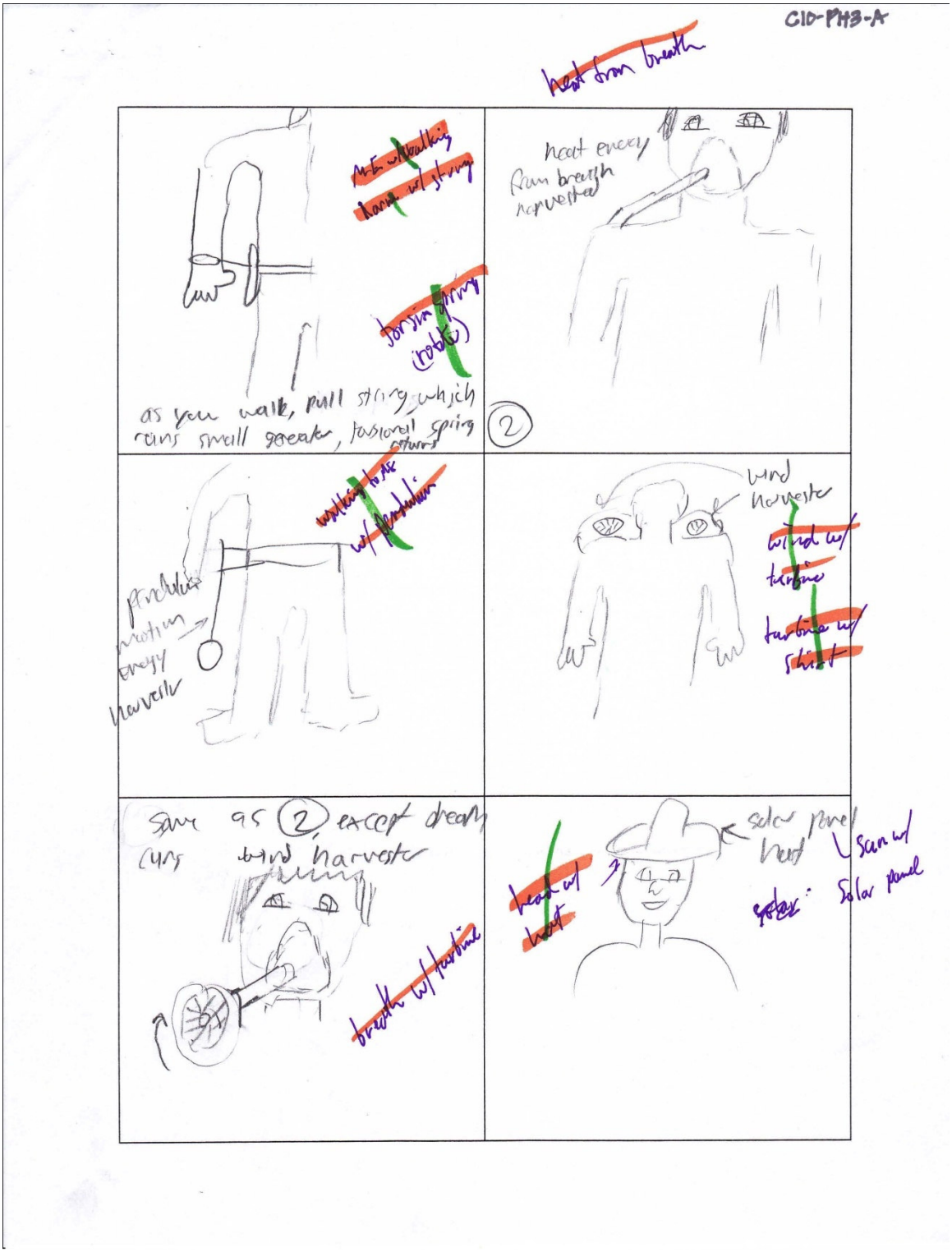
C9-PH3-B

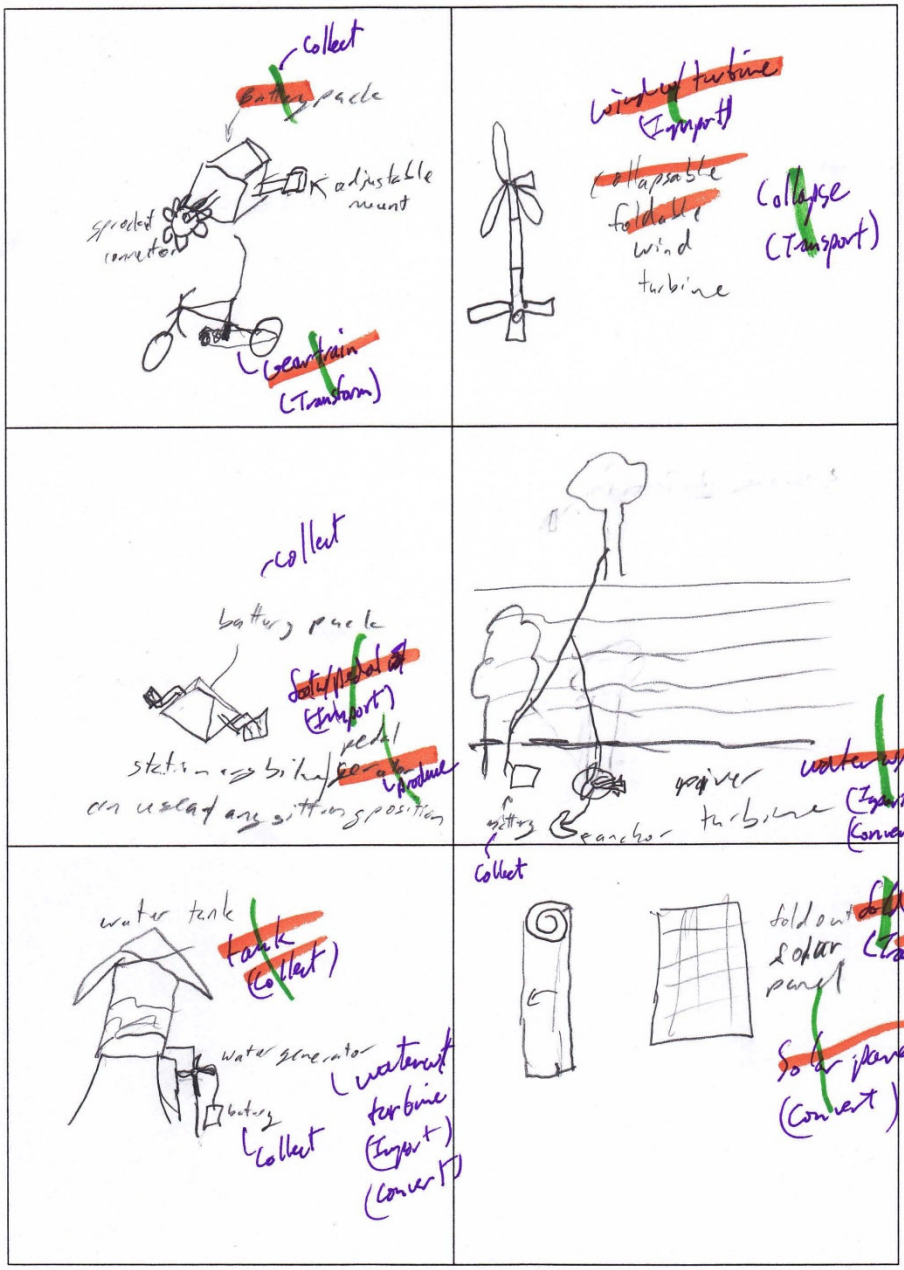
attach device to their EYELIDS so it captures the energy from their blinking!! (ME from blinky constellate)	Put a whirligig (bidirectional) in their wind pipe to collect energy from their breathing. breath w/ wind mill
Combine all ideas: 	


foot w/shoe
(Import)
compression plates in the shoes
leg w/ pants
(Import)
foot w/pedals
(Import)
Heat Engine
(Convert)
Human w/ vest
(Import)
Convert
(Transport)
(Collect)

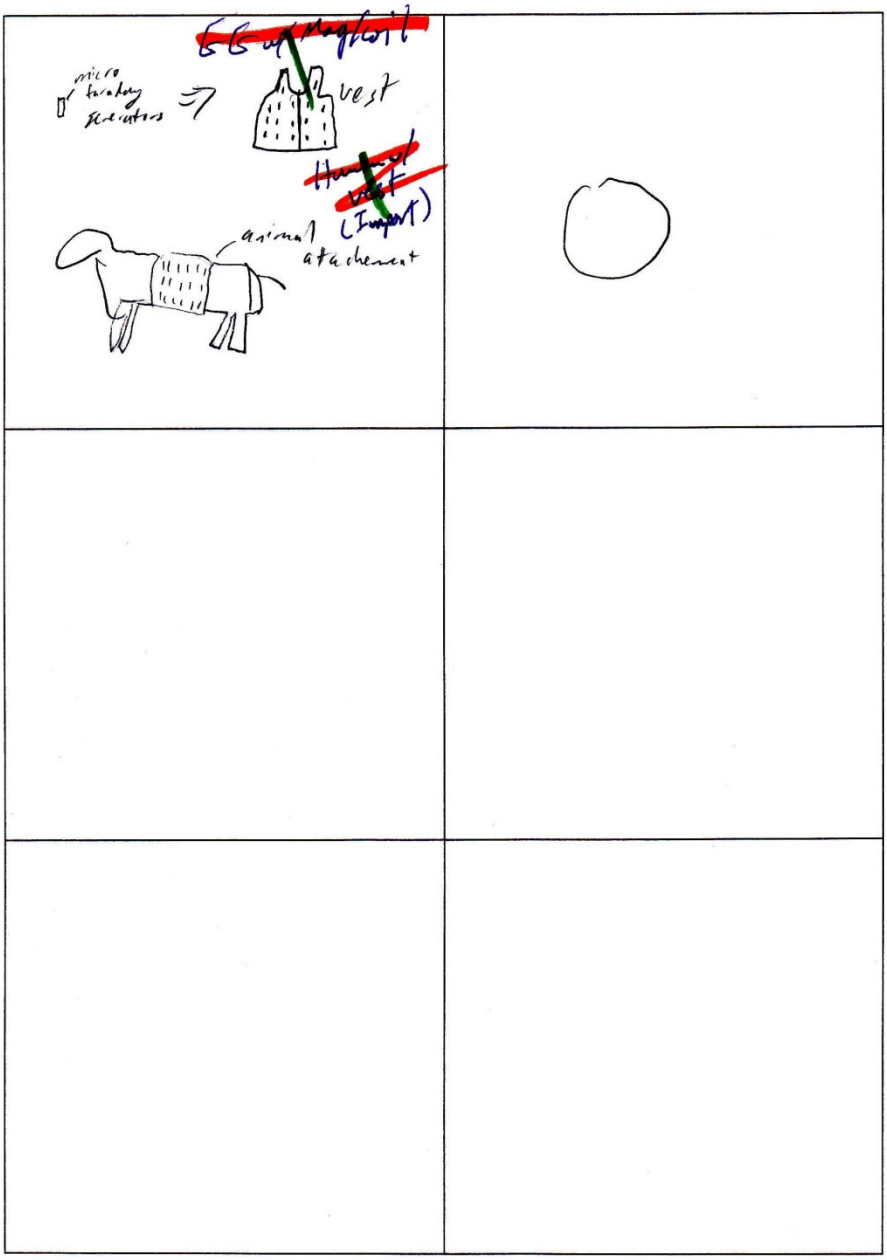
C10-PH1-B



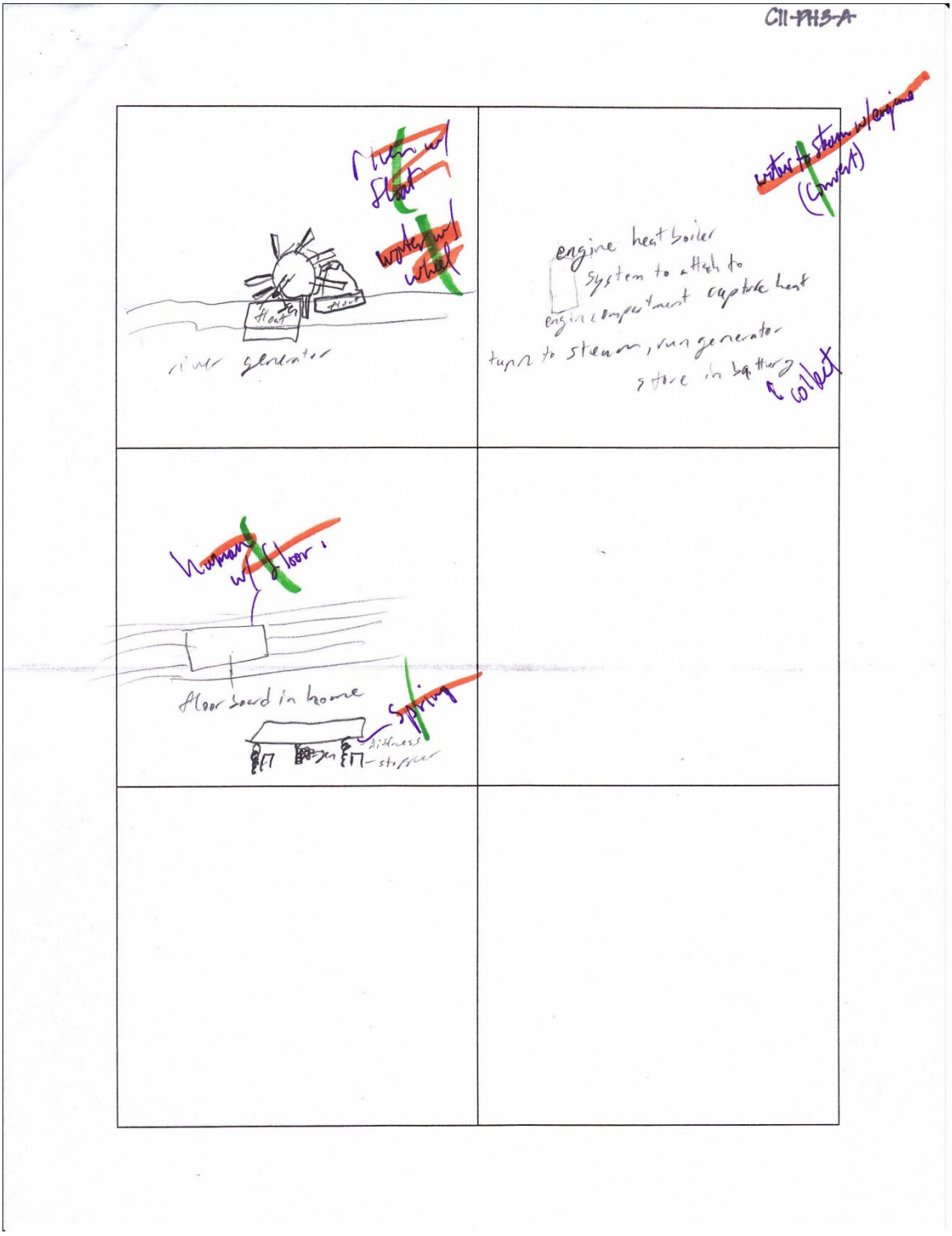

C10-PH3-A
heat from breath
as you walk, pull string which runs small speaker, torsional spring
heat energy from breath harvester
2
wind harvester
same as 2 except dream runs wind harvester
solar panel
heat w/ heat
scan w/ solar: Solar panel


collect
adjustable mount
(transform)
collapse
(transport)
collect
(collect)
(convert)
collect
(transport)
(convert)

C11-PH1-B



C11-PH5-A

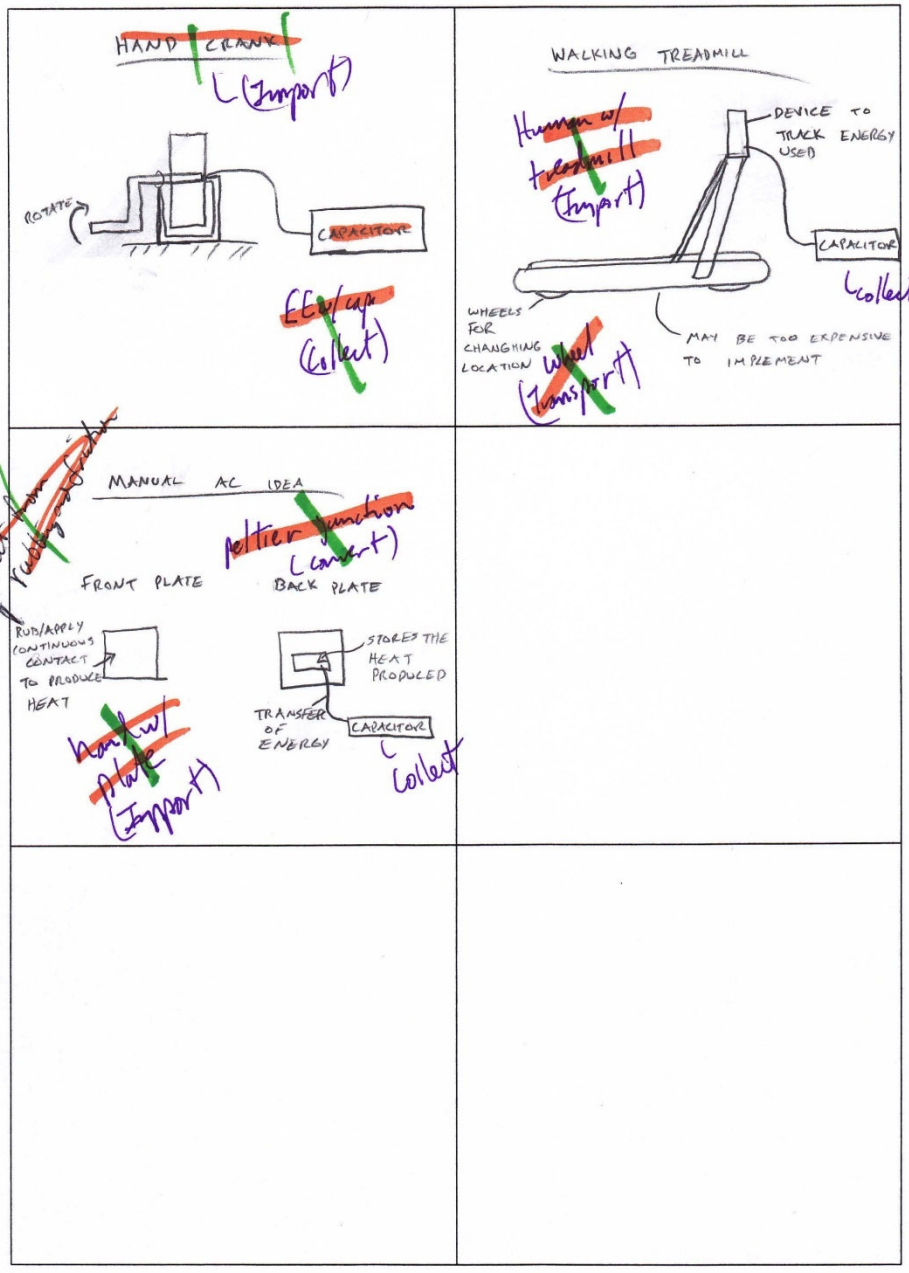

River generator
engine heat boiler
System to attach to
engine compartment capture heat
turn to steam, run generator
store in battery
collect
floor sound in home
spring
stiffness

C12-PH1-A

Bicycle / pedal a bike to generate electricity	Running treadmills — similar concept as bicycle
use mechanical pumps that can transport water & generate electricity	use animals Create motion on a wheel with hamsters/gerbils Combining these can provide adequate electricity at low cost

~~ME w/ gas engine~~ A simple 2stroke engine can be effective for small community. low maintenance & installation cost	~~wind w/ windmill~~ incorporate the wind; if some areas of Africa are windy. A simple windmill can be used. ~~water w/ containers~~
The rotary water wheel is a good idea, but this combined with a small engine would create effciency & increase power output. ~~water w/ water wheel~~	Storage box for water. when it fills, a mechanical system dumps it. the velocity of water drives a motor to generate electricity. This allows you to control the velocity of water. Could even use same water, repeatedly. ~~pumping water (expert)~~ ~~water w/ turbine~~ ~~EE w/ motor~~

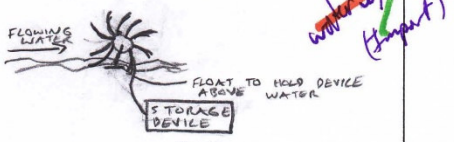
CB-PHI-A


HAND CRANK
(Import)
ROTATE
CAPACITOR
Eff/cap
(Collect)
WALKING TREADMILL
Human w/ treadmill
(Import)
DEVICE TO TRACK ENERGY USED
CAPACITOR
Collect
WHEELS FOR CHANGING LOCATION
(Transport)
MAY BE TOO EXPENSIVE TO IMPLEMENT
MANUAL AC IDEA
peltier function
(convert)
FRONT PLATE
BACK PLATE
RUB/APPLY CONTINUOUS CONTACT TO PRODUCE HEAT
heat w/ plate
(Import)
STORES THE HEAT PRODUCED
TRANSFER OF ENERGY
CAPACITOR
Collect

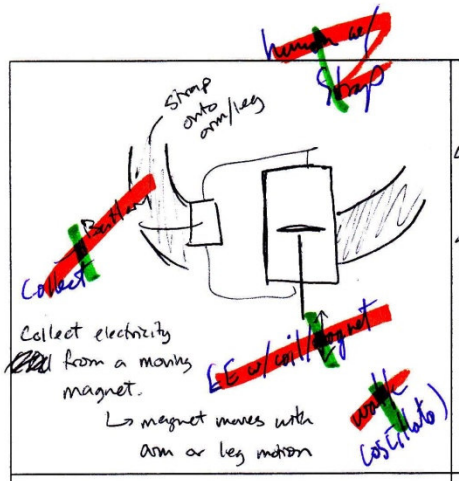
C13-PHB-A

HAND CRANK IDEA
INCORPORATED WITH
THE WATER WHEEL

SIDE VIEW

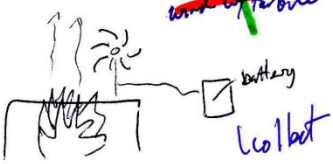


CH4-PH1-A



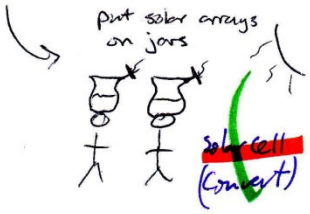
Sorry not human-powered

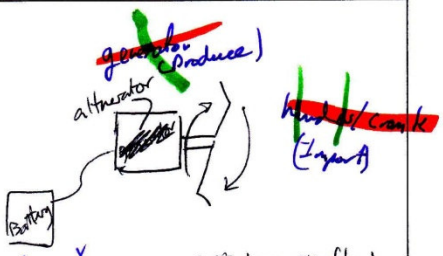
as you cook let natural convection turn a small windmill



Collect

from what I know, many women have to carry water from the source to the village all day.

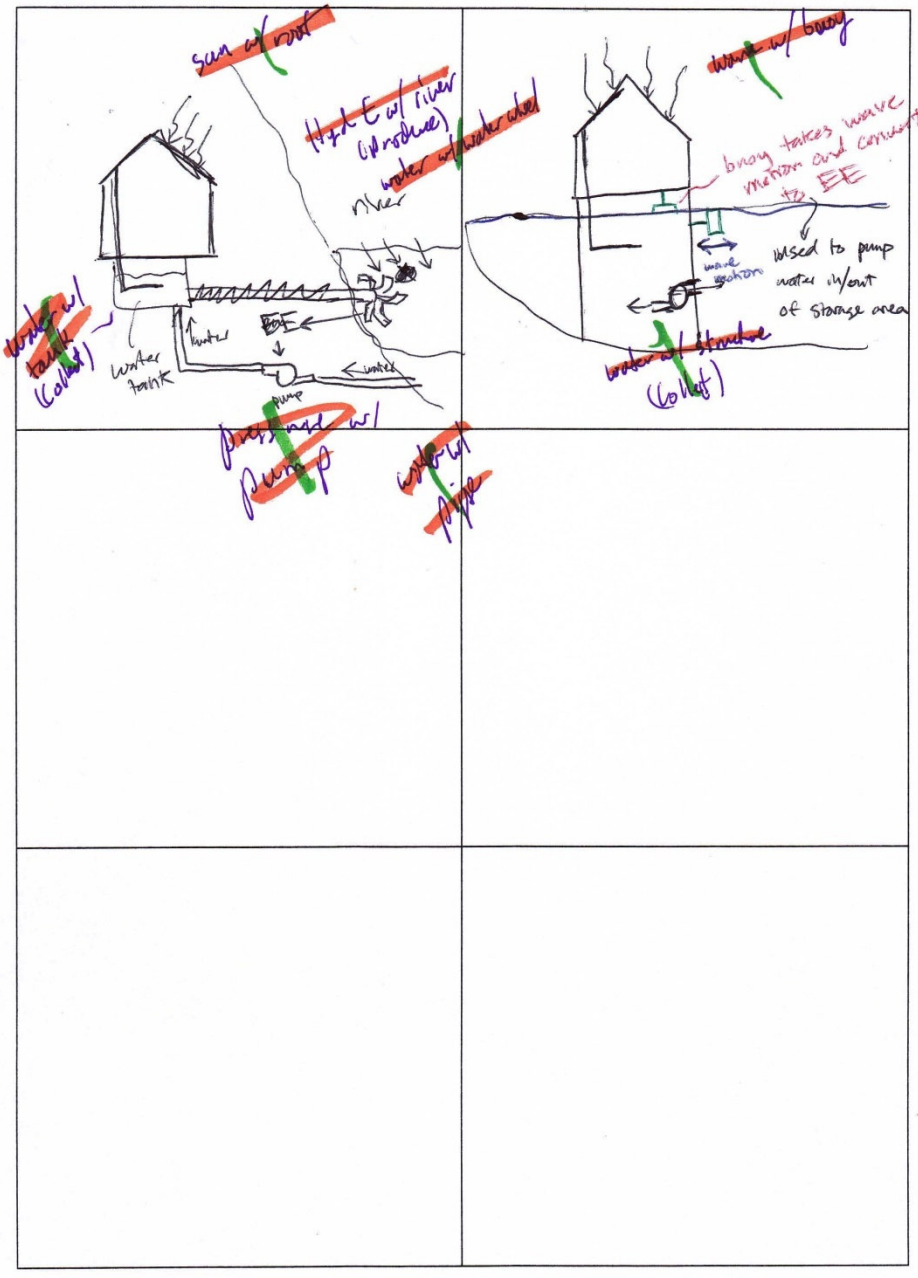




Collect

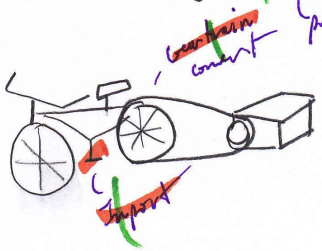
just turn shaft to produce current

C14-PH3-A

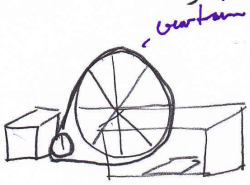

sun w/ wave
Hydro E w/ river (produce)
water w/ water wheel
river
water w/ tank (collect)
water tank
water
pump
pump used w/ pump
water w/ pipe
wave w/ buoy
buoy takes wave motion and converts to EE
used to pump water in/out of storage area
wave motion
water w/ structure (collect)

C15·PH1-A

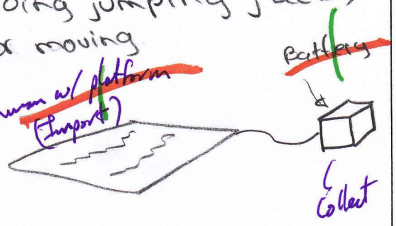
Bicycle with back "wheel" attached to a generator



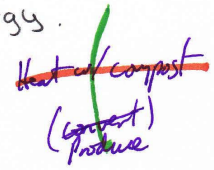
Sewing mill wheel, used while making clothes.



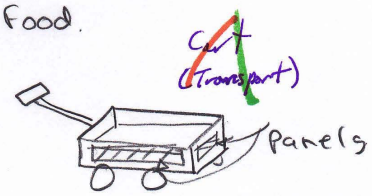
Platform to harvest human motion while doing jumping jacks or moving



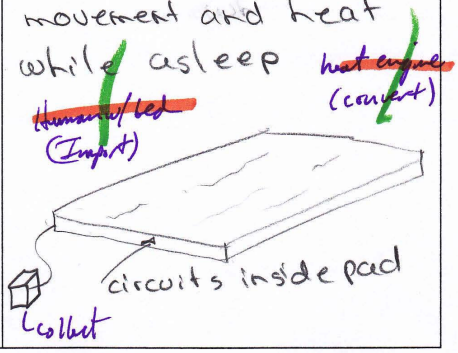
Compost material used to generate energy. Food left overs to be stored and harvest energy.



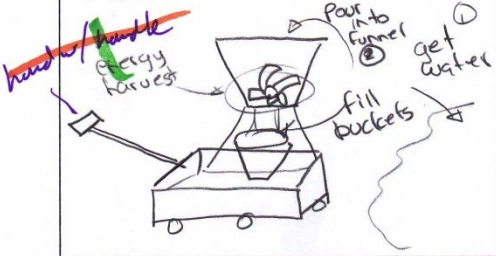
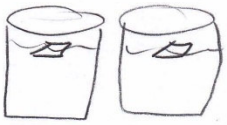
Cart with small solar panels. Cart to be used to transport water or food.

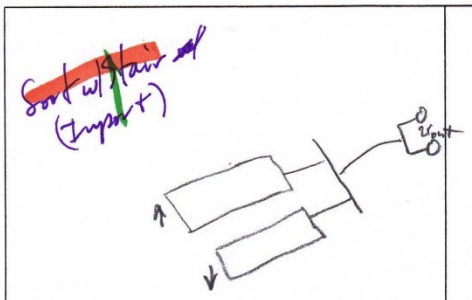


Bed mats to harvest movement and heat while asleep

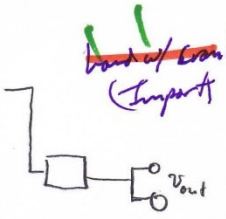


C15-PH3-A

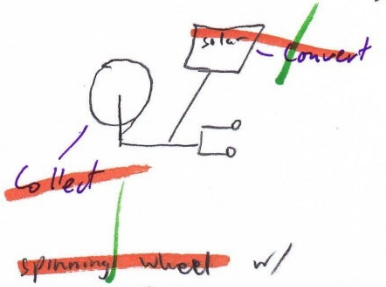
Device used on top of cooking fire filled with water to generate heat energy 	
A cart used to transport buckets of water, fitted with a funnel with a motor at the end 	water w/ cart (transport) water w/ funnel (Import) water w/ wok-wheel water w/ bucket (Collect) water from lake (produce)
The containers where they store their water could use water movement harvest technology 	waves from bucket (produce) wave w/ float (Convert)



Step machine

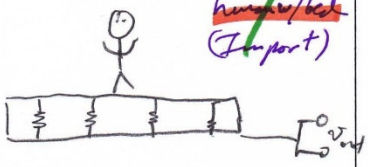


Crank



Collect

spinning wheel w/ solar collector



in heel store energy

oscillate

(not portable though)

(oscillate)

Something placed between two points on leg that stores energy when walking

Something that stores the energy from swinging arms.

C16-PH1-B

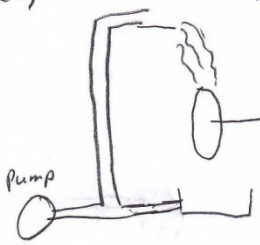
human w/ clothes  convert A solar collector outfit (not from human motion)	A toy for children to play with that stores energy Part of toy (collect)
something that can store energy from the vibrational energy of cattle.	

CIV-PH3-A

(1)

humans/cattle walks over plate that stores energy

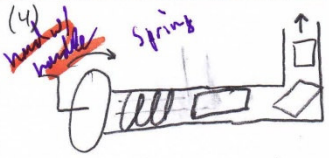
(2)



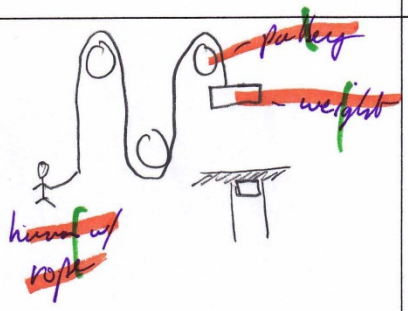
manual pump that helps recirculate water that turns a wheel

(3) combine (1) and (2) and solve for energy

(4)



a piston that goes down a groove when turned, subsequently released and provides "one shot" of energy

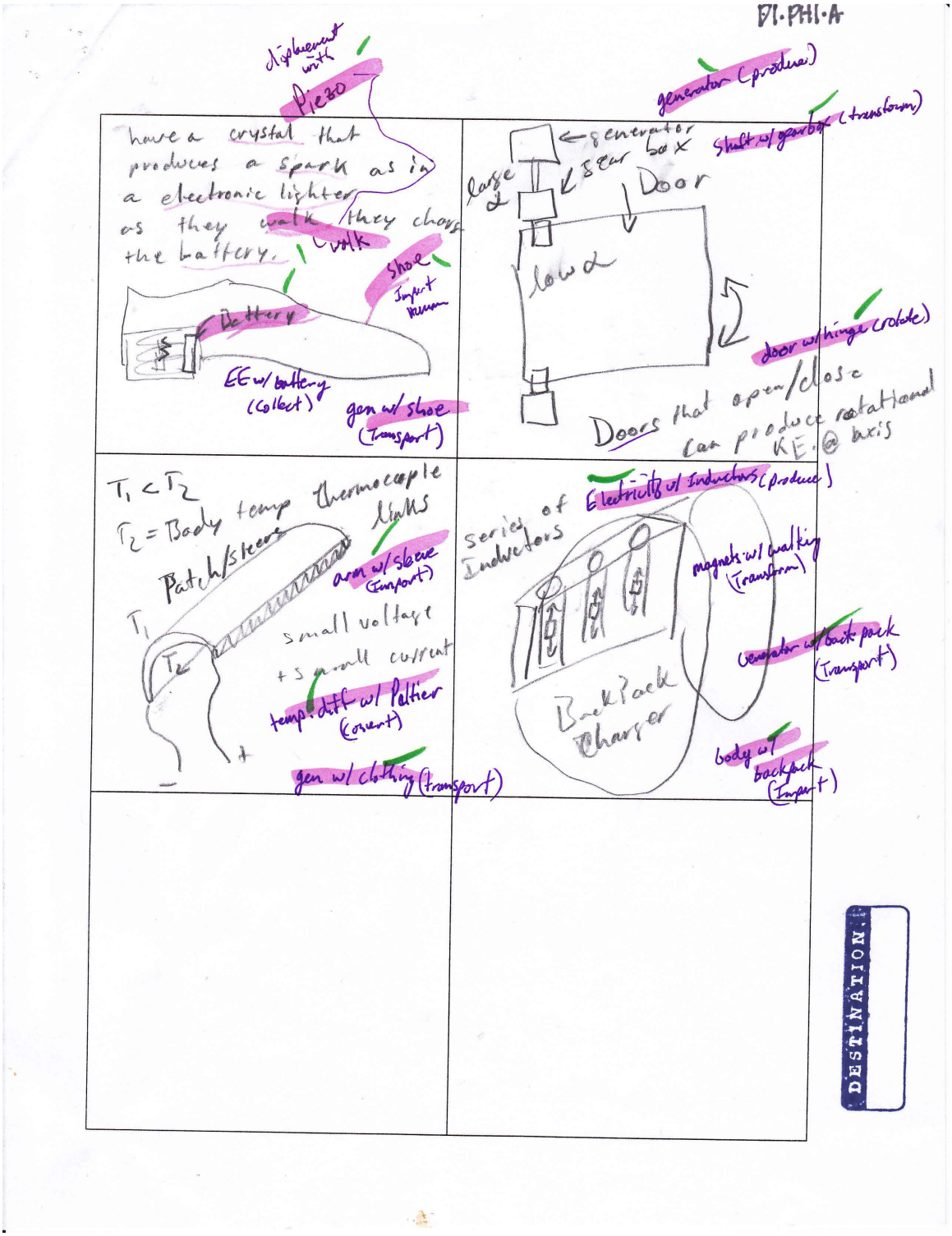


PLACE RODS UNDER STAIR STEPS AND COUPLE THESE TO A CRANKSHAFT. AS PEOPLE WALK UP/DOWN THE STEPS, THE CRANKSHAFT RUNS A GENERATOR.	PLACE A WATER WHEEL OVER A RIVER. AS WATER FLOWS, THE WHEEL TURNS AND POWERS A GENERATOR.
USE SOME SORT OF BIMETALLIC MECHANISM TO GENERATE POWER. THE MOTION OF THIS STRIP WILL BE INITIATED BY THE EXCESS HEAT BEING RELEASED WHILE COOKING OR IN HEATED ROOMS.	USE GYM BIKES TO GENERATE POWER SIMILAR TO THE METHOD OF A HAND CRANK.

PLACE RODS UNDER STAIR STEPS AND COUPLE THESE TO A CRANKSHAFT. AS PEOPLE WALK UP/DOWN THE STEPS, THE CRANKSHAFT RUNS A GENERATOR.	PLACE A WATER WHEEL OVER A RIVER. AS WATER FLOWS, THE WHEEL TURNS AND POWERS A GENERATOR.
USE SOME SORT OF BIMETALLIC MECHANISM TO GENERATE POWER. THE MOTION OF THIS STRIP WILL BE INITIATED BY THE EXCESS HEAT BEING RELEASED WHILE COOKING OR IN HEATED ROOMS.	USE GYM BIKES TO GENERATE POWER SIMILAR TO THE METHOD OF A HAND CRANK.

USE A SYSTEM SIMILAR to the Stair TO THE STAIR-STEPPING GENERATOR, BUT SUBSTITUTE HYDRAULIC ENERGY FOR MECHANICAL ENERGY. ~~pressure w/ stair piston~~	PLACE A DEVICE SIMILAR TO THE WAVE OPERATED POWER APPARATUS IN ~~the~~ PERSONAL SWIMMING POOLS. ~~wave w/ swimming pool~~

GROUP D


DI·PHI·A
displacement with
Piezo
have a crystal that produces a spark as in a electronic lighter as they walk. they charge the battery.
walk
shoe
Impact Vacuum
battery
EE w/ battery (collect)
gen w/ shoe (Transport)
generator (produces)
generator box
shaft w/ gearbox (transform)
large gear
Door
door w/ hinge (rotate)
Doors that open/close can produce rotational KE @ axis
$T_1 < T_2$
T_2 = Body temp
Thermocouple
lights
Patch/sleeve
arm w/ sleeve (Transport)
small voltage
+ small current
temp. diff w/ Peltier (convert)
gen w/ clothing (transport)
Series of Inductors
Electricity w/ Inductors (produce)
magnet w/ walking (Transform)
generator w/ back pack (Transport)
Back Pack Charger
body w/ backpack (Input)
DESTINATION

DI·PHS·A

ME w/ Ball

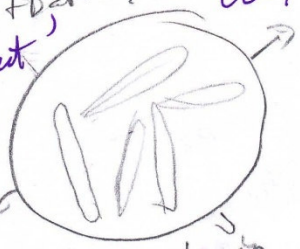
Balanced Soccer Ball

with inductors inside

+Battery

EE w/ Induction

collect



Soccer is popular in most other countries

hand w/ knife

machete / generator

Battery & generator

moving mass inside tube

mass in tube (move)

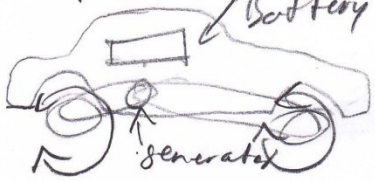
every swing forces the mass out and a wound spring brings it back. (used during harvests

oscillate

"Toy" Car

kid plays with it + charges battery

Battery



generator

add noise for fun

ME w/ toy

Swing set

rotating shaft for energy

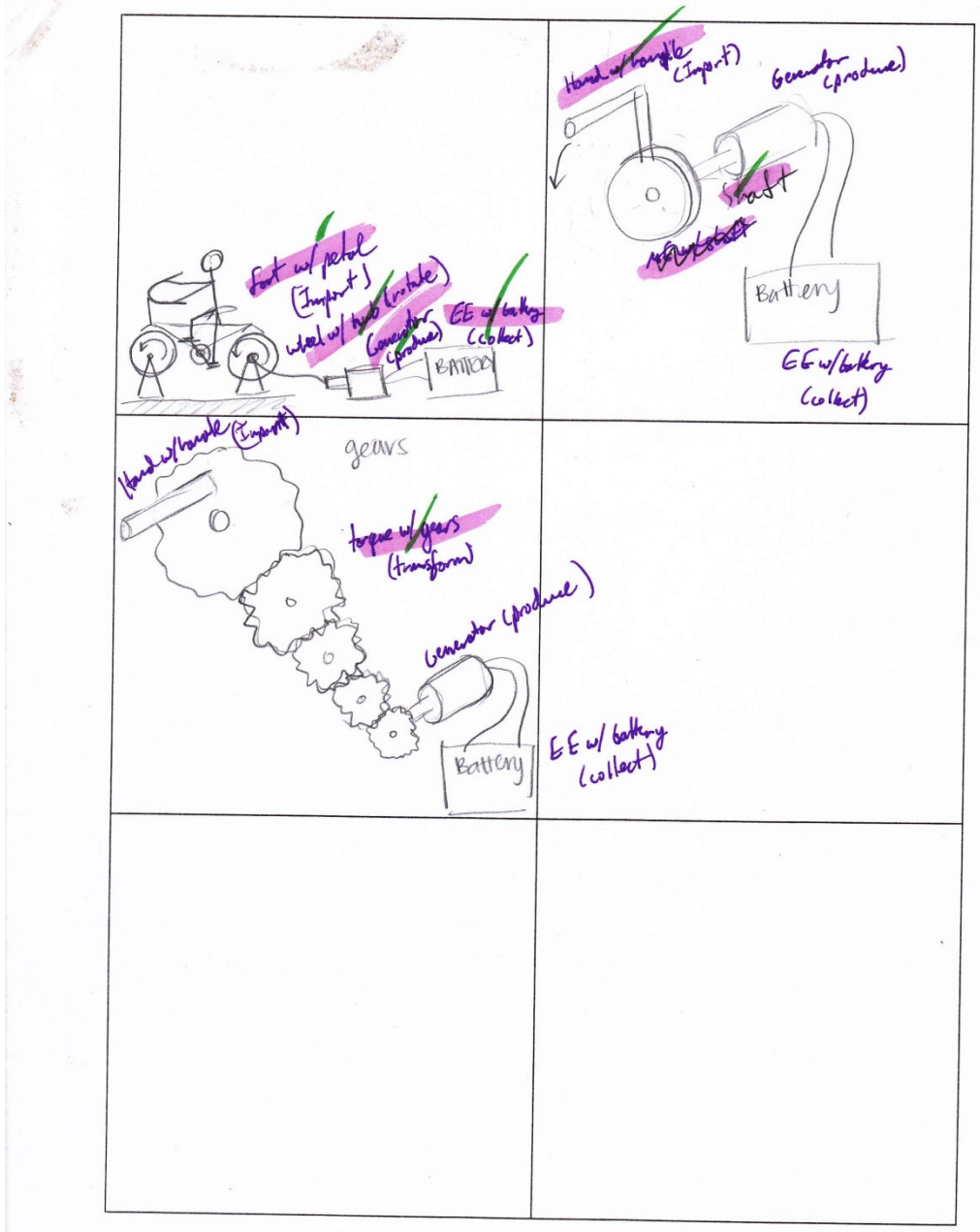
shaft (rotate)

human w/ swing

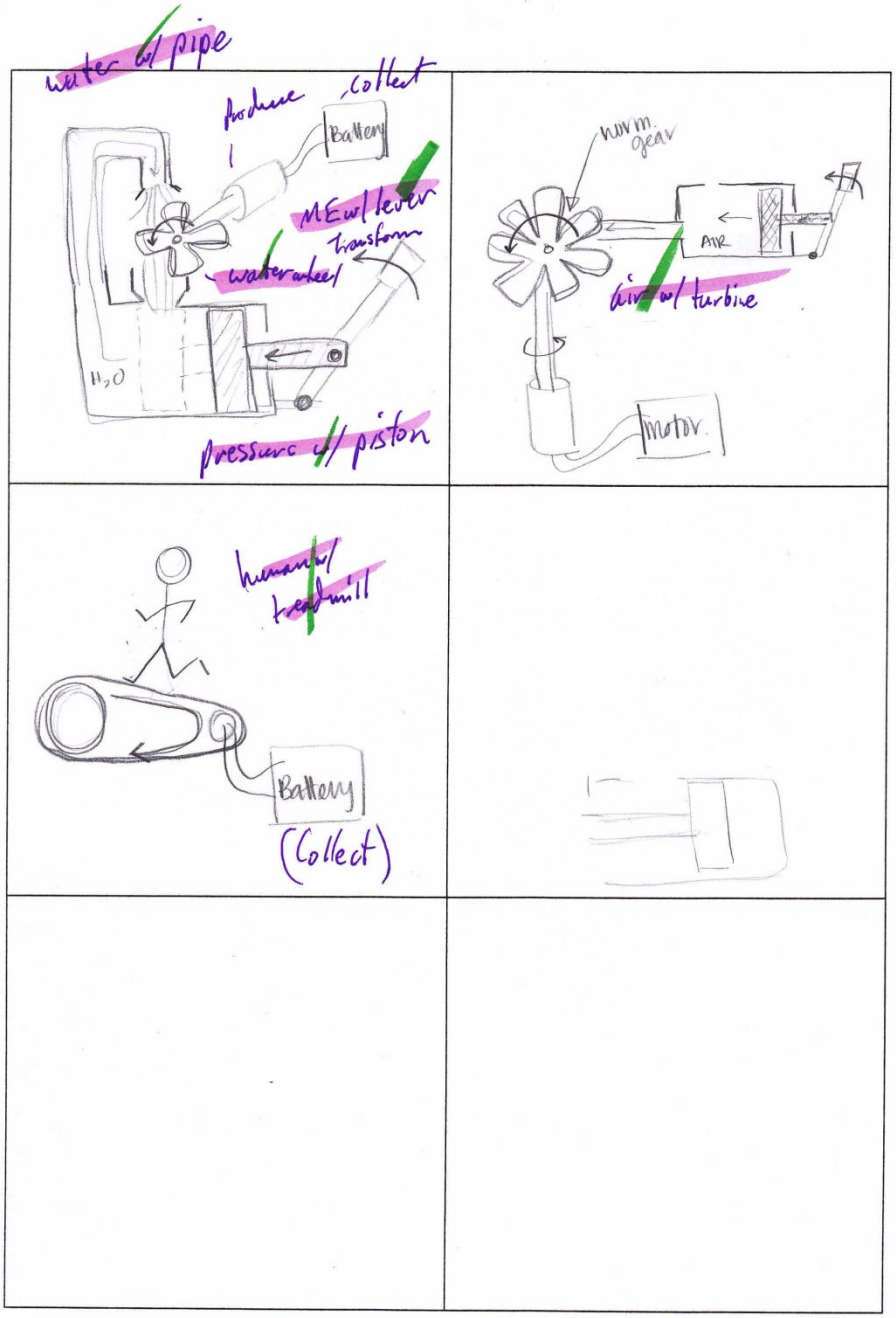
generators

human w/ seesaw

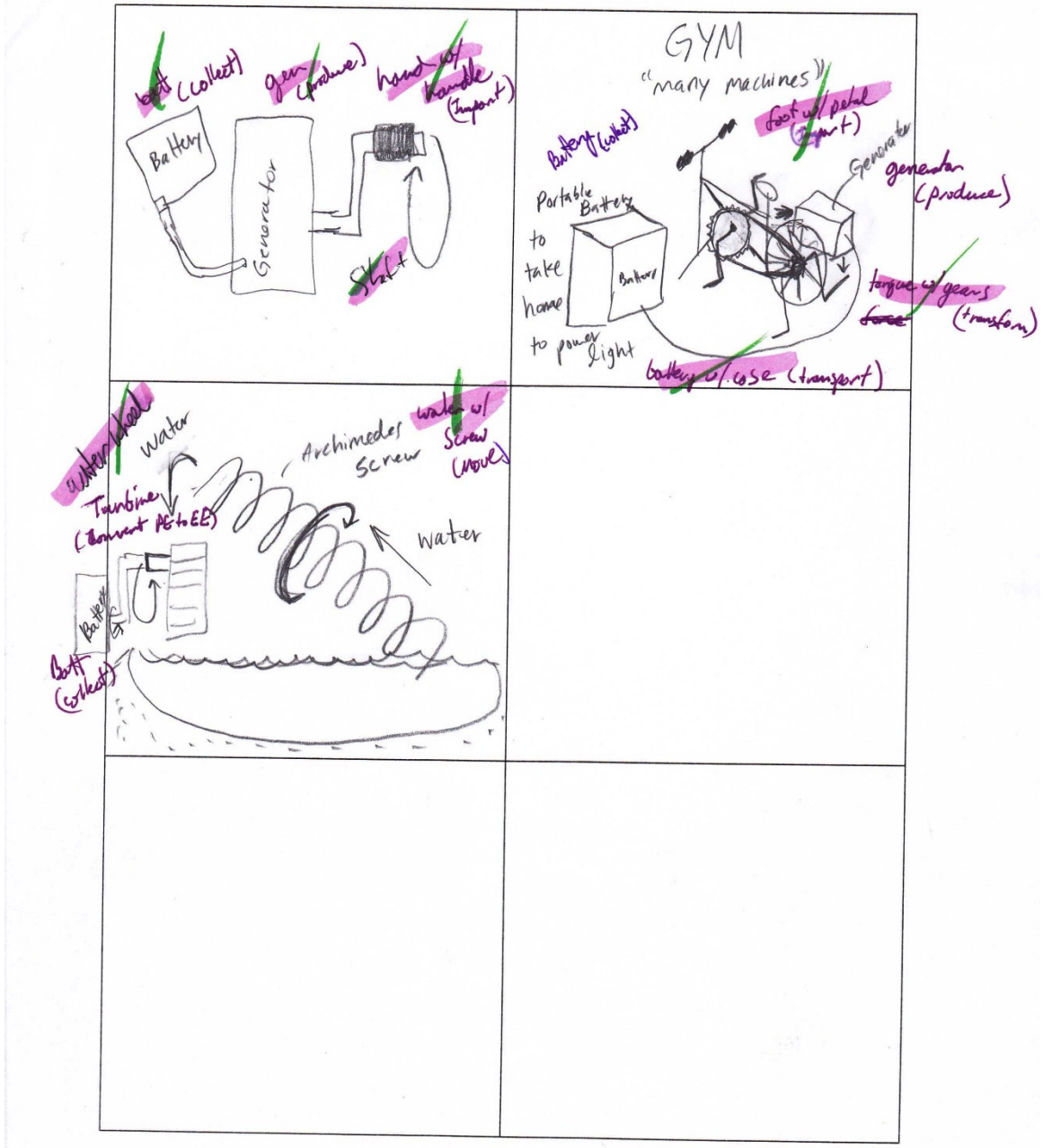
D2·PH1·A


foot w/ pedal (Input)
wheel w/ hub (rotate)
Generator (produce)
EE w/ battery (collect)
BATTERY
Hand w/ handle (Input)
Generator (produce)
Shaft
Battery
EE w/ battery (collect)
Hand w/ handle (Input)
gears
torque w/ gears (transform)
Generator (produce)
Battery
EE w/ battery (collect)

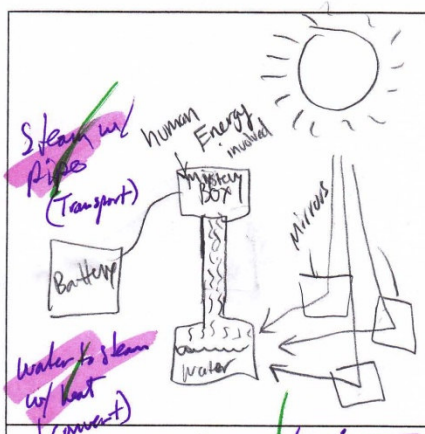
D2-PH3-A

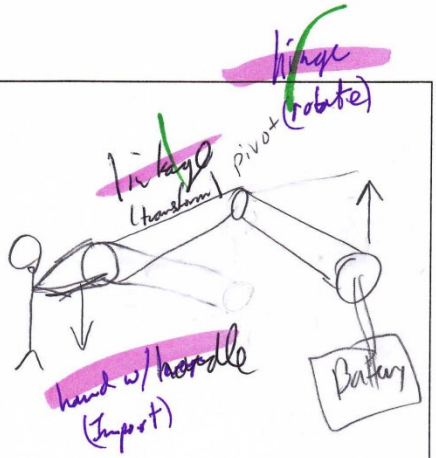

water w/ pipe
produce
collect
Battery
ME w/ lever
transform
water wheel
H_2O
pressure w/ piston
worm gear
AIR
air w/ turbine
motor
human w/ treadmill
Battery
(Collect)

D3.PHI.A


Battery
Generator
Skate
GYM
"many machines"
Portable Battery
to take home to power light
Battery
Generator
Water
Archimedes Screw
Water
Battery

D3·PH3·A


Steam w/ Pipe
(Transport)
Human Energy input
Battery Box
Battery
mirrors
water
Water to steam w/ heat
(convert)
Sun w/ mirror
(Import)


linkage
(rotate)
linkage
(transform)
pivot
hand w/ handle
(Import)
Battery

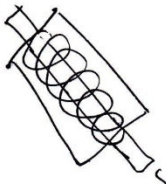
D4 PHI A

Bicycle that charges removable battery; will help them transport to work & will create energy.

Energy can be created from wheel rotation. Or a mini wind turbine placed in front.

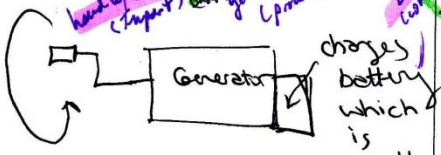
"Shake - for - energy"

Using an induction principle the person shakes the device to store energy



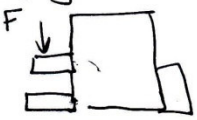
can be worn or shaken by hand.

"Crank" Device

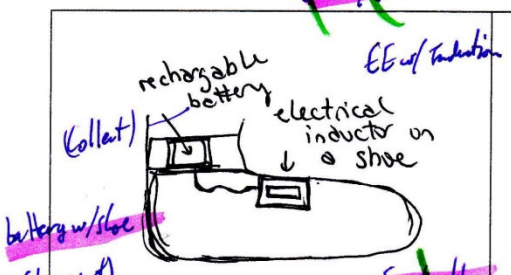
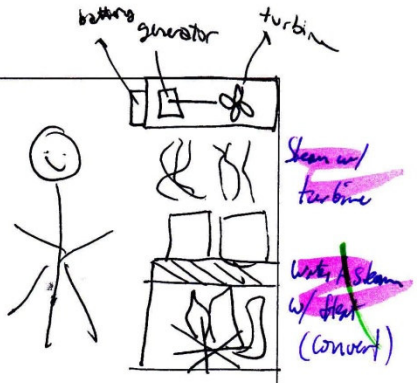


Solar Cell Backpack w/ charger inside

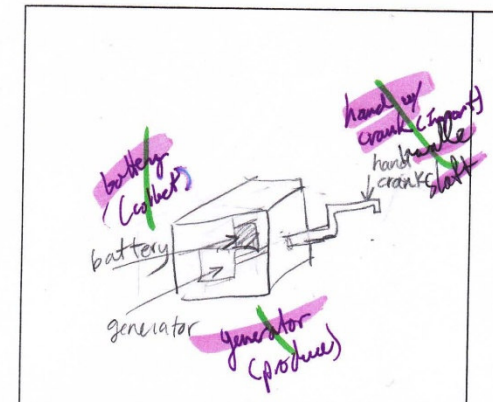
Small Stair Stepping device that transfers force from your foot/weight to charge a removable battery



D4.PH3.A

 motion from walking charges the battery	 Steam collector from a human cooking
Add Solar power to the bike (might make product too expensive)	

D5.PH.A

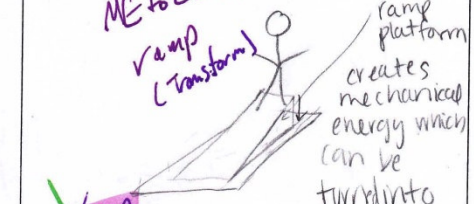


hand w/ handle (Import)

pressure w/ pump (produce)

pump like bike pump, creates hydraulic E to be transformed into EE

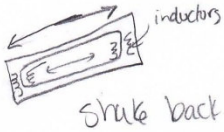
hydraulic to Electrical (Transform)



ME to EE w/ ramp (Transform)

creates mechanical energy which can be turned into electrical

body w/ ramp (Import)



EE w/ Inductors (produce)

Shake back & forth to create EE thru inductors

Shake (oscillate)

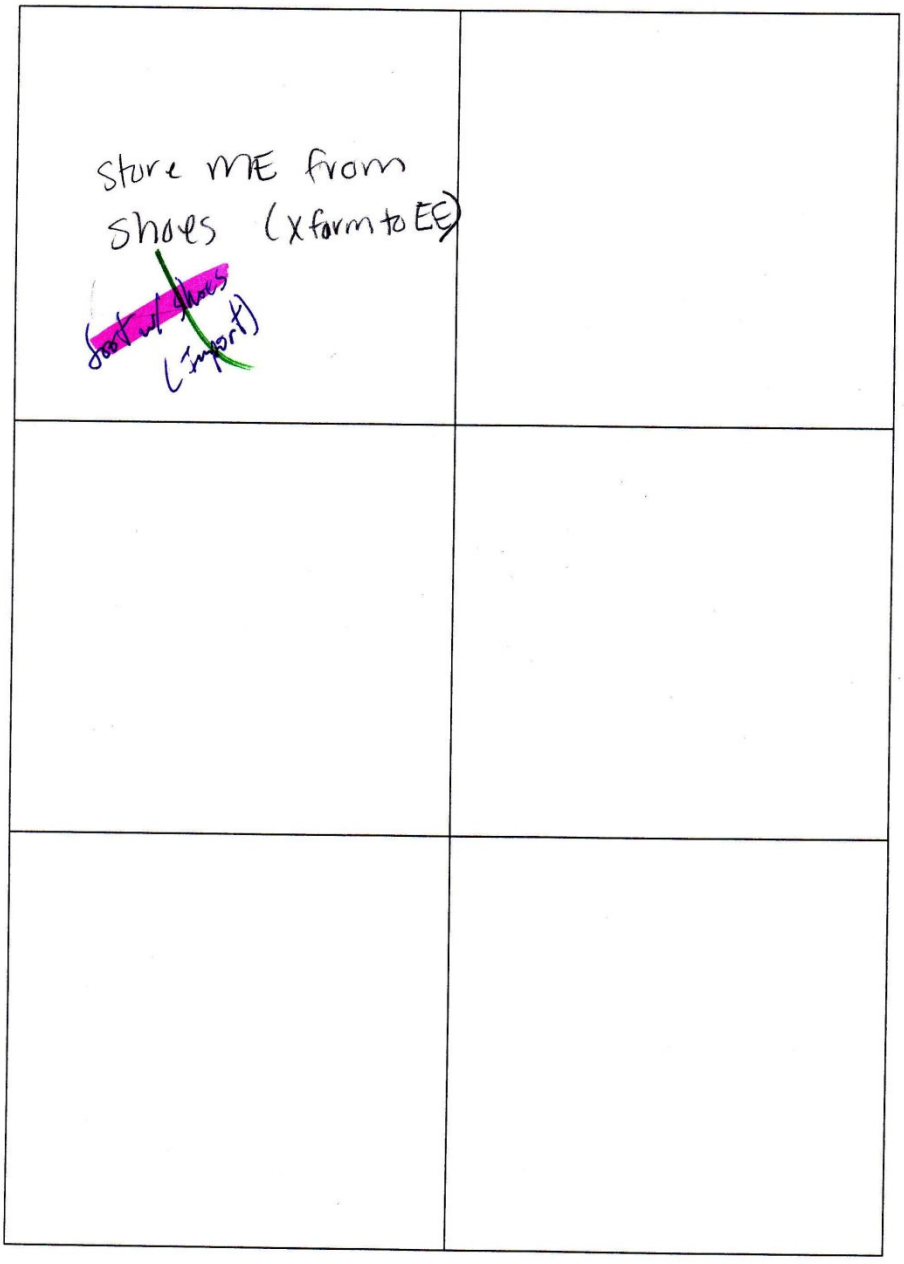
backpack that harnesses vibrational energy.

back pack (Transport)

ME w/ Toy (collect)

put something inside toys to capture mechanical energy

D5·PHI·B



D5·PHS·A

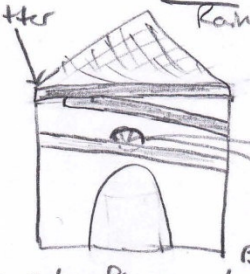
human w/ pogo stick pogo-stick transform energy from spring into EE spring oscillate	human w/ chair spring  chairs w/ seat on springs that collapse when someone sits
use body heat as thermal energy ΔT from body (produce)	miniature steam generator  steam w/ heat
have a hydro-generator that water goes through any time its poured into a jug or bottle (water w/ bottle	have filtration system that also generates energy from hydropower

D6·PHI·A

Rain w/ gutters (Input)

Source: Rain water

Gutter



Paddle wheel (convert)

Battery (store)

• Rainwater flowers down a gutter and rotates a paddle w/ a small generator (generator (produces))

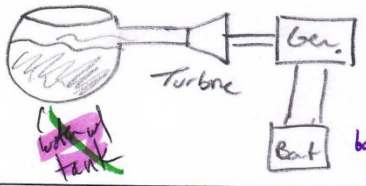
Heat from burning wood (Transform)

Source: Heat from wood burning fires (e.g. wood burning stoves); outputs steam which drives a turbine

Steam w/ heat (Convey)

Steam w/ heat (Convey)

Turbine (Input (Convert))



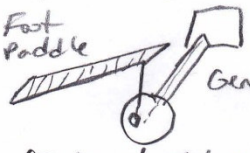
generator (produces)

battery (collect)

water w/ tank

• Foot-depression lever generating electricity (similar to an old-fashioned sewing machine) ← analogy

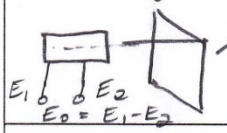
Foot pedal (Input)



Crank-and-slider (crank slider mechanism (Rotate))

EE w/ Induction

- Magnetic induction to charge a battery
  - > A magnet attached to a rod on a door induces a current as it slides in and out of the magnetic field

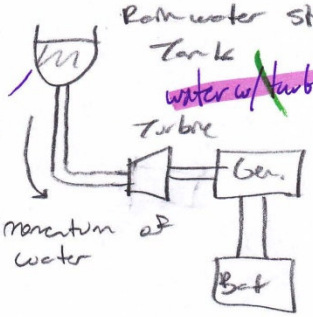


Magnet w/ door

- Source: Livestock
  - Donkey/Cattle is attached to a lever which rotates a gear train, spinning a generator, charging a battery

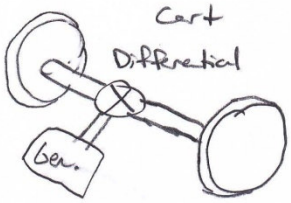
animal w/ lever
Convert
produce
collect

Source: Rain water



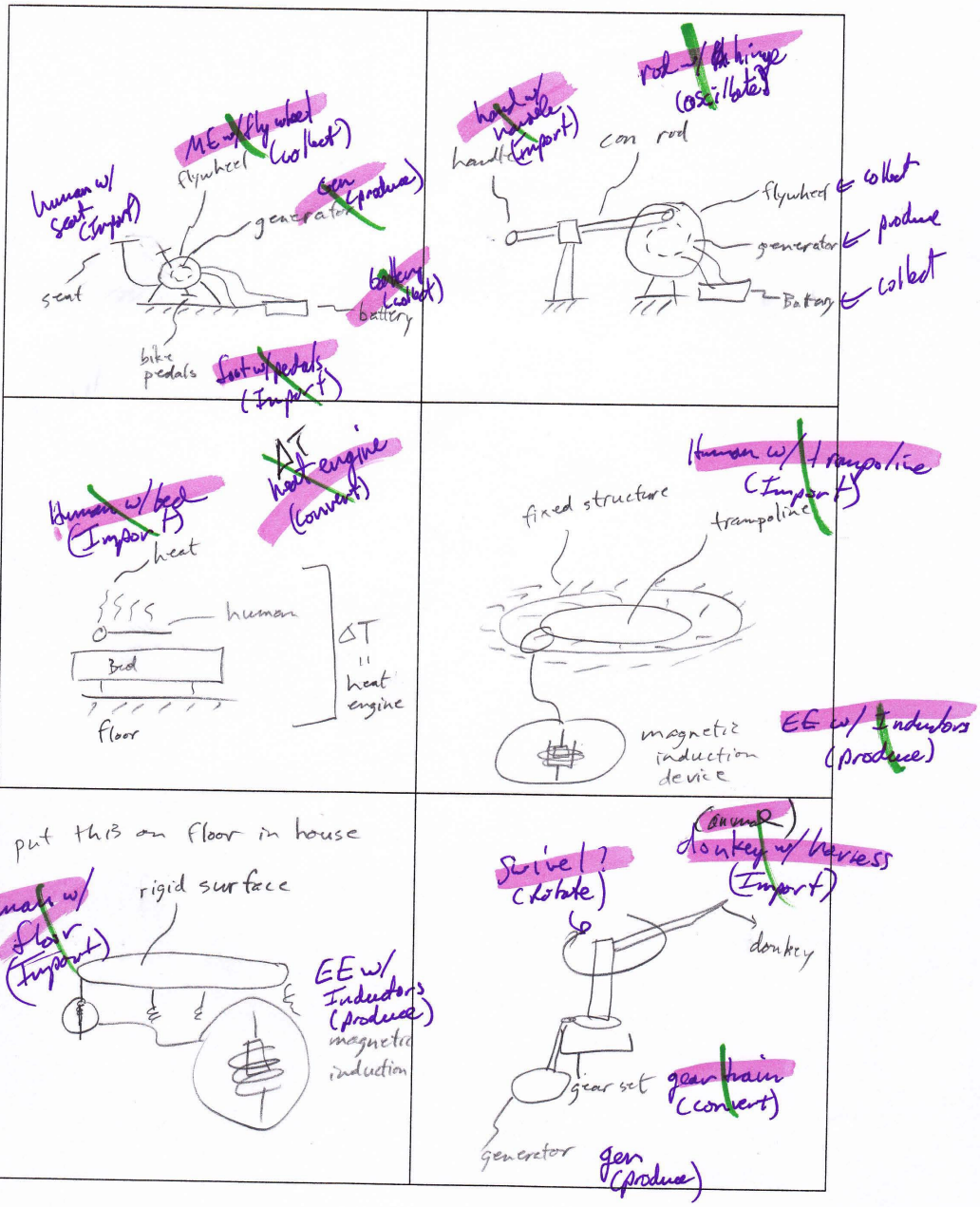
water w/ tanks (Collect)

Sources: ~~Shaft~~ rotate work from spinning Cart

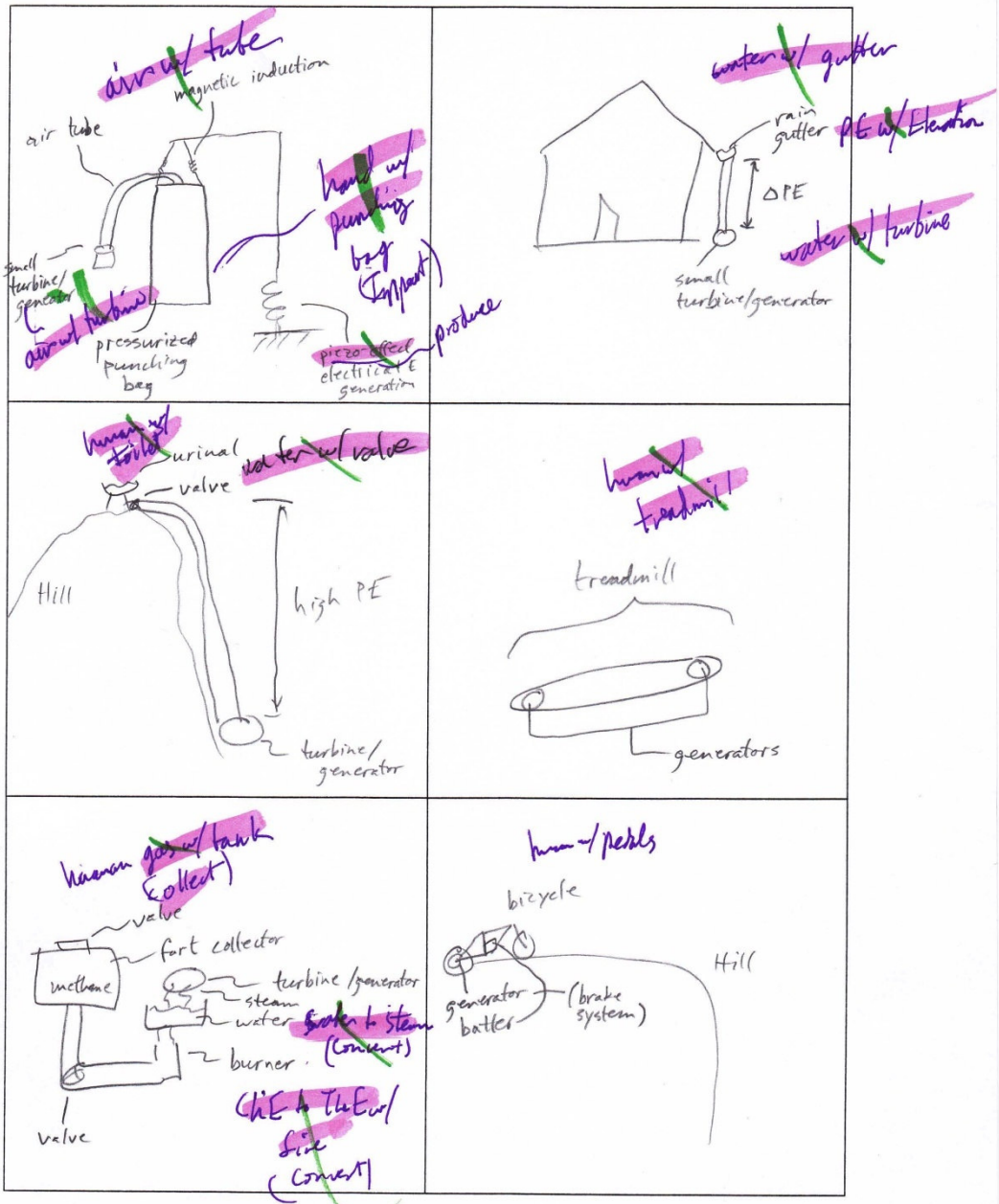


Source: Excess food stuffs

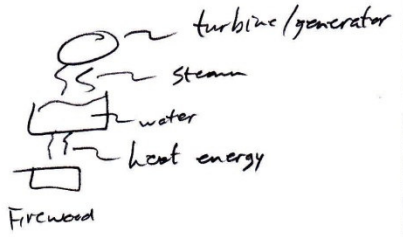
D7·PHI·A


human w/ seat (input)
ME w/ flywheel (collect)
flywheel
gen (produce)
generator
seat
battery
bike pedals
foot w/ pedals (input)
hand w/ handle (input)
con rod
flywheel
generator
Battery
collect
produce
collect
human w/ bed (input)
heat
human
bed
floor
ΔT = heat engine
fixed structure
trampoline
magnetic induction device
EE w/ inductors (produce)
put this on floor in house
rigid surface
human w/ floor (input)
EE w/ Inductors (produce)
magnetic induction
donkey w/ harness (input)
donkey
gear set
gear train (convert)
generator
gen (produce)

D7·PH3·A

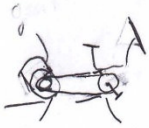

air tube
magnetic induction
small turbine/ generator
pressurized punching bag
piezo-effect electricity generation
rain gutter
ΔPE
small turbine/generator
urinal
valve
Hill
high PE
turbine/ generator
treadmill
generators
valve
fart collector
methane
turbine/generator
steam
water
burner
valve
bicycle
Hill
generator battery
(brake system)

D7·PH3·B



DS·PHI·A

bicycle system

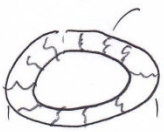


Human w/ Beat (Import)

foot w/ pedal (Import)

springs collect energy

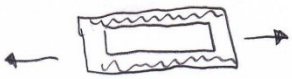
Trampoline



Human w/ trampoline (Import)

PE w/ springs (collect)

Fun for Kids!

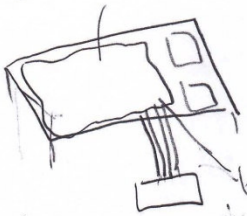


shake to charge

Shake (oscillate)

working fluid in blanket w/ low boiling pt. ammonia?

boil fluid (convert)



human heat w/ blanket (Import)

blanket

absorbs heat from people, keeping them cool, energy converted to electric!

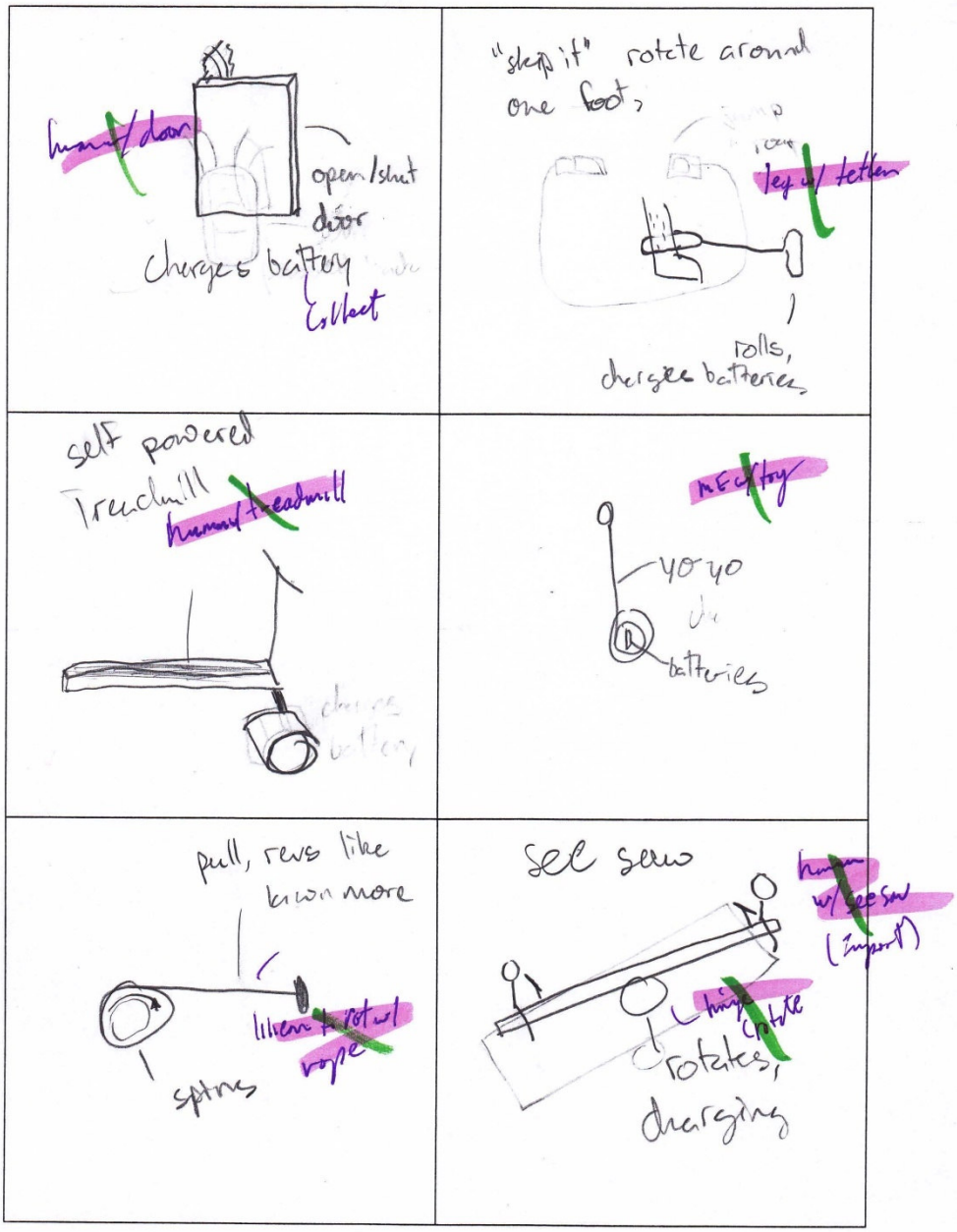
wind w/ windmill (Import)

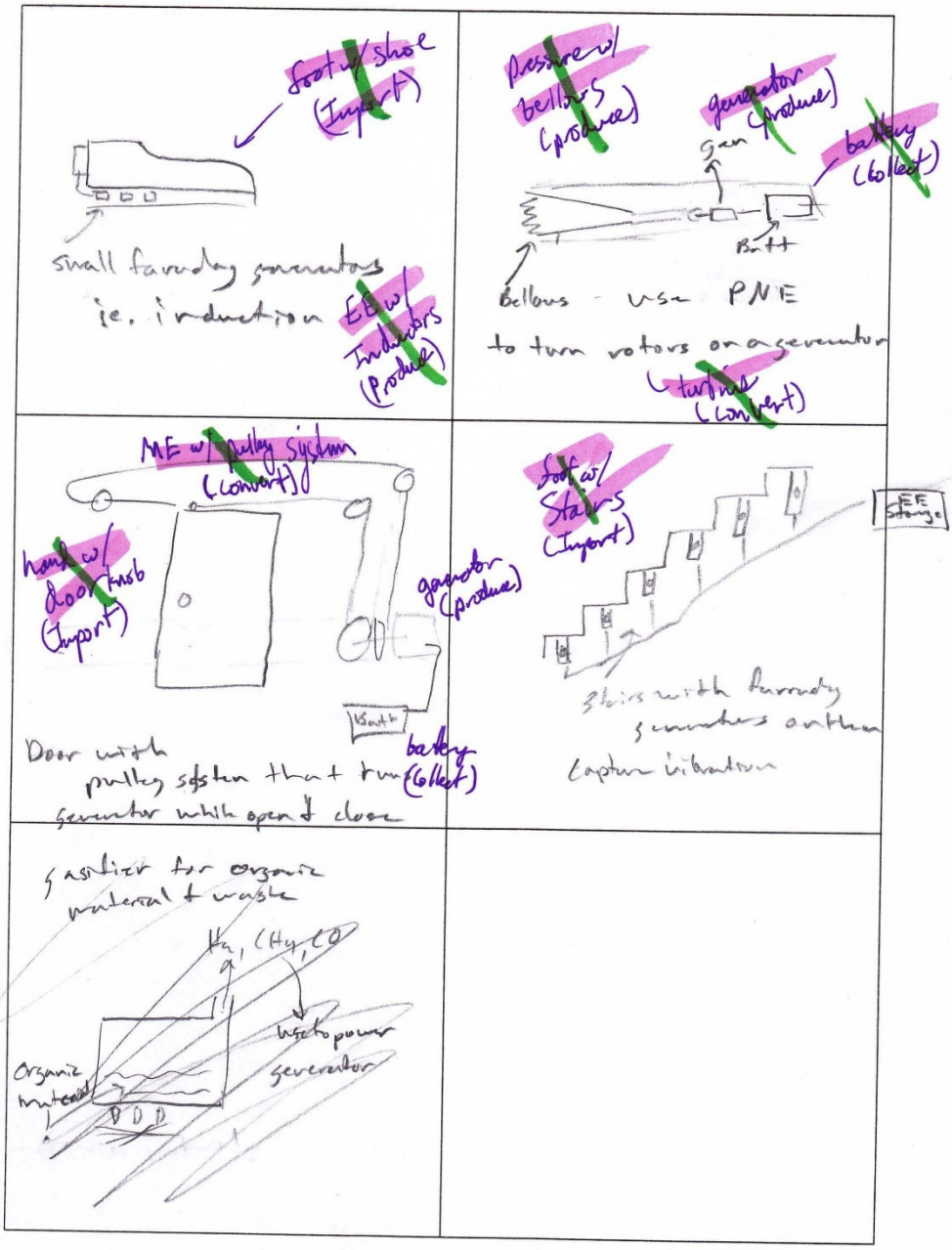
hand w/ handle (Import)

"wind mill" carried by person / attached to backpack, charges batteries

battery (collect)

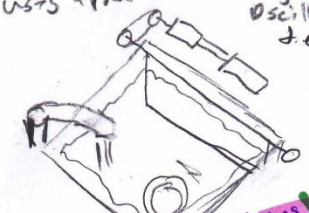
D8·PH3·A


human/door
open/shut door
charges battery
Exhaust
"step it" rotate around one foot,
leg w/ tether
rolls,
charges batteries
self powered Treadmill
human/treadmill
charges battery
MEchToy
40 40
batteries
pull, revs like known more
human/rotor/rope
springs
see saw
human w/ seesaw (import)
hinge
rotates,
charging


foot w/ shoe (Input)
small faraday generators ie. induction
EE w/ Inductors (Produce)
Pressure w/ bellows (produce)
generator (produce)
gas
battery (collect)
Batt
Bellows - use PNE to turn rotors on generator
turbine (convert)
ME w/ pulley system (convert)
home w/ door knob (Input)
generator (produce)
Batt
battery (collect)
Door with pulley system that turn generator while open & close
foot w/ Stairs (Input)
EE storage
Stairs with faraday generators on them Capture vibration
gasifier for organic material & waste
H2, CHy, CO
Organic material
heat to power generator

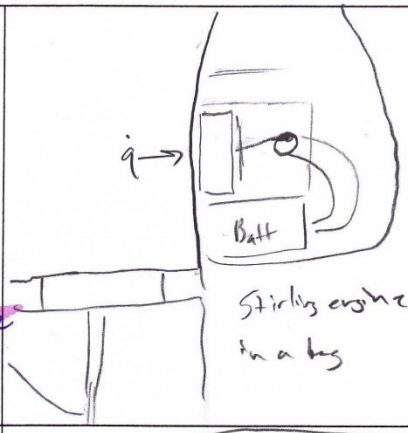
D9-PH3-A

Paddle or Paddles in sink or washing trough uses tidal motion to generate Oscillations & electricity



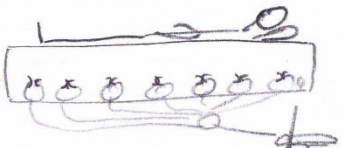
water w/ sink

waves w/ paddle (oscillate)



Stirling engine in a bag

Movement in bed forces air from airsacks powering a small turbine

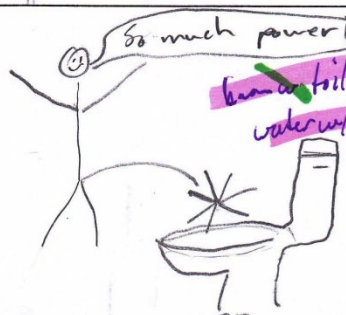


human w/ bed

pressure of bladder

air w/ turbine

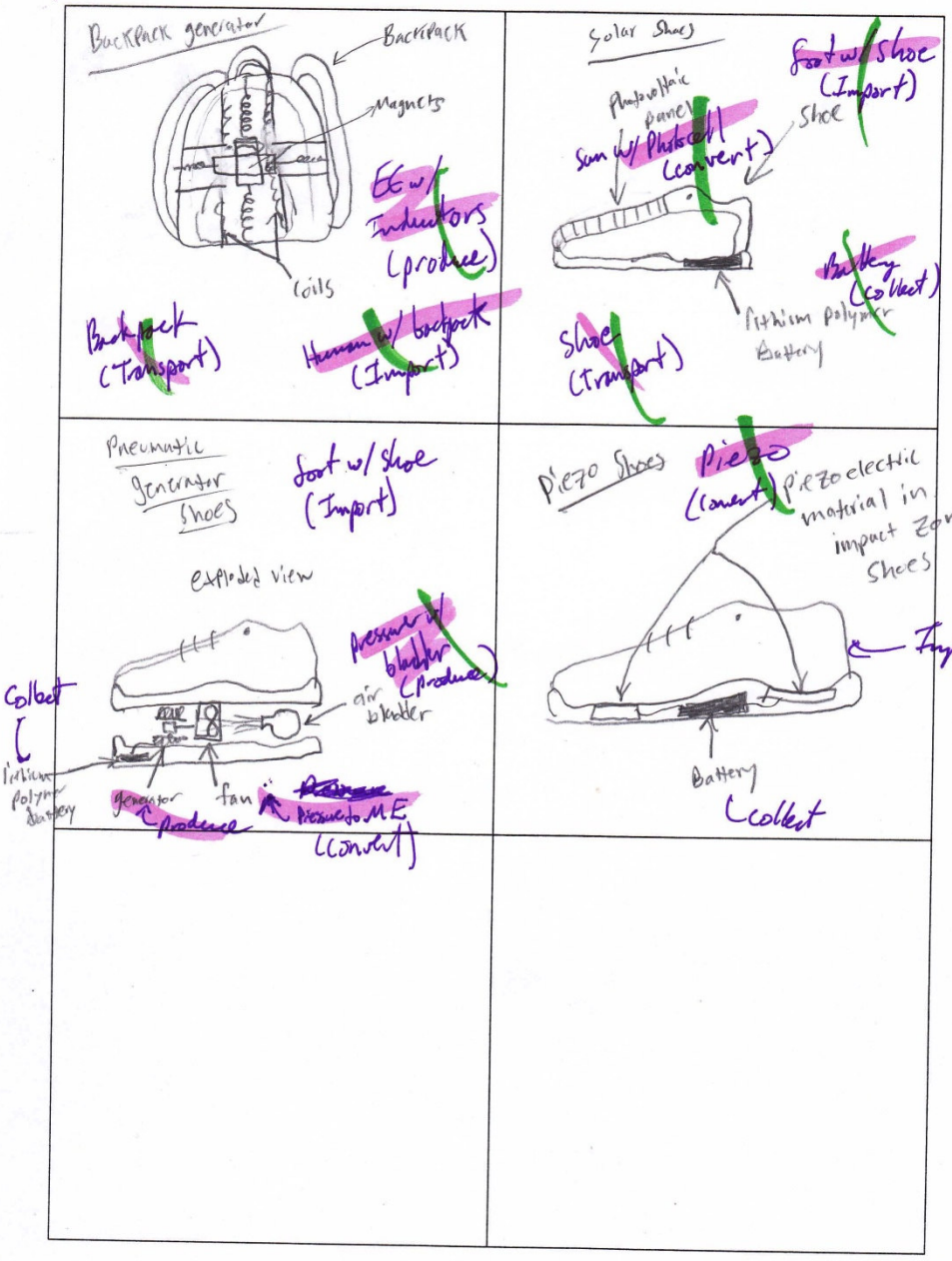
Windmill



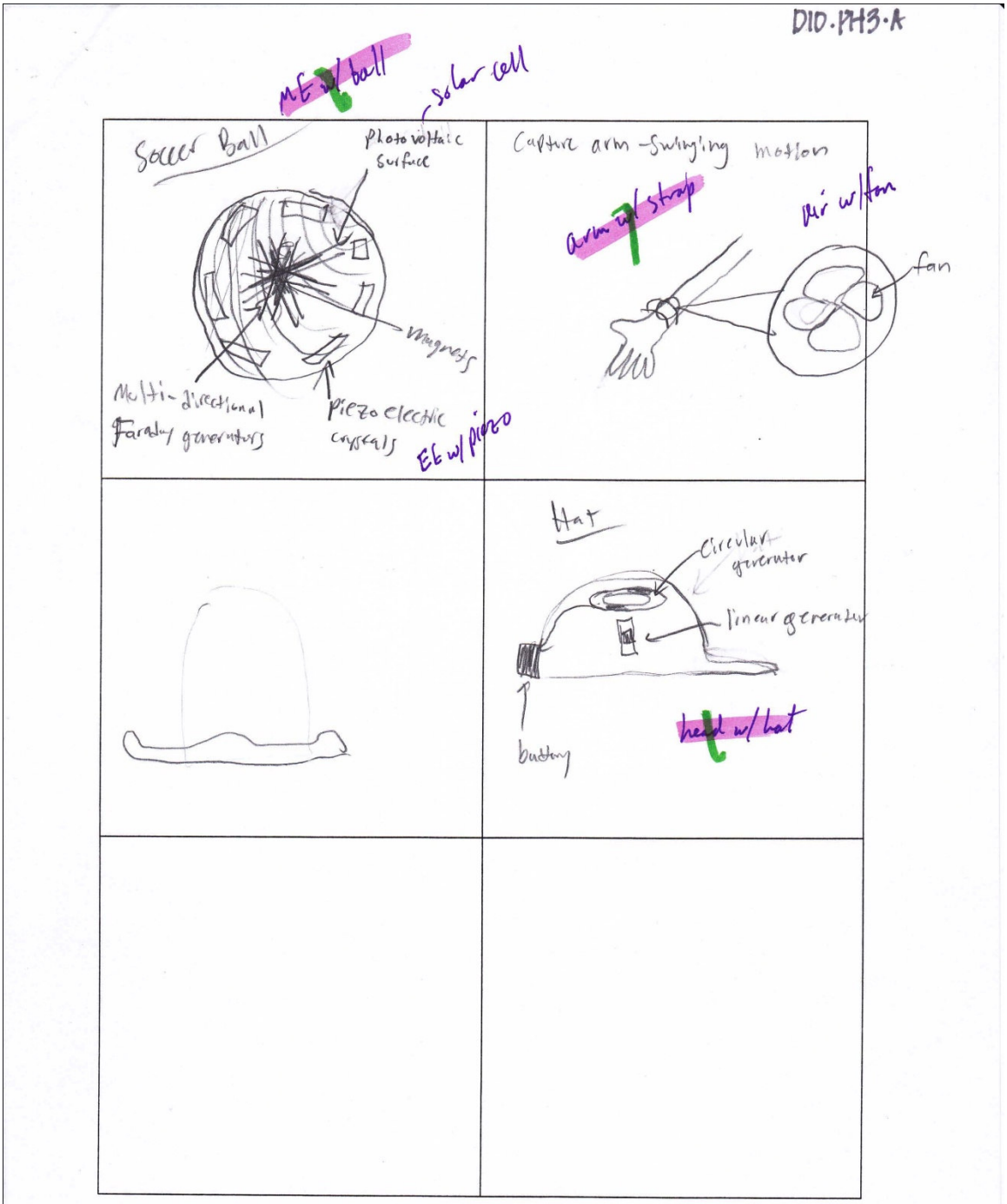
human w/ toilet

water w/ water wheel

Wee - Wheel © generate elec from pee stream


Backpack generator
Backpack
Magnets
Coils
EE w/ Inductors (produce)
Backpack (Transport)
Human w/ backpack (Import)
Solar Shoes
Photovoltaic panel
Sun w/ Photocell (convert)
Shoe
foot w/ shoe (Import)
Battery (collect)
Lithium polymer Battery
Shoe (Transport)
Pneumatic Generator Shoes
foot w/ shoe (Import)
Exploded view
pressure in bladder (produce)
air bladder
Collect
lithium polymer battery
generator
Produce
fan
Pressure to ME (convert)
Piezo Shoes
Piezo (convert)
piezoelectric material in impact zones of shoes
Import
Battery
Collect

DID.PH3.A


ME w/ ball
solar cell
Soccer Ball
Photovoltaic surface
Magnets
Multi-directional Faraday generators
Piezo electric crystals
EE w/ piezo
Capture arm - Swinging motion
arm w/ strap
air w/ fan
fan
Hat
Circular generator
linear generator
battery
head w/ hat

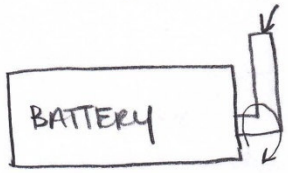
D11·PH1·A

REVOLVE CRANK TO CREATE ENERGY WHICH WILL BE STORED IN THE BATTERY.

handle w/ crank (Import)

collect

CRANK



(THIS HAS BEEN DONE W/ FLASHLIGHTS)

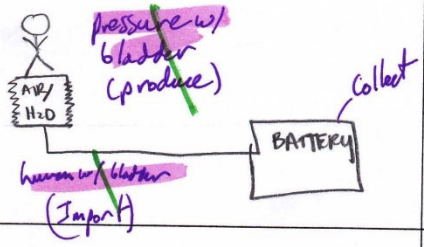
backpack (Transport)

WHILE WALKING AROUND THESE RURAL AREAS, USERS CAN PUT SHOES/BACKPACK ON AND PUMP AIR INTO BACKPACK WHICH HAS MECHANISM TO CONVERT THE PUMPED AIR INTO EE TO STORE IN BATTERY

foot w/ shoe (Import)

pressure to E (convert)

USER CAN FILL A PLASTIC HOLDING COMPARTMENT W/ AIR / H_2O & USE HANDS/ JUMP TO COMPRESS H_2O / AIR TO APPLY PRESSURE TO BATTERY WHICH CHARGES IT.



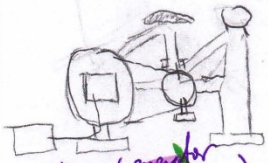
D11·PHS·A

USE THE HUMAN FORCE IN CHEWING TO CHARGE A BATTERY BY CREATING A POSITIVE & NEGATIVE VOLTAGE (WHEN THE PERSON CHEWS) TO CREATE CURRENT THRU BATTERY	DEVELOP PADS WHICH USERS CAN SIT ON TO TRANSLATE FORCE → EE TO CHARGE BATTERIES.

D12·PH1·A

Mod kit to an existing bike; pedal powering a generator

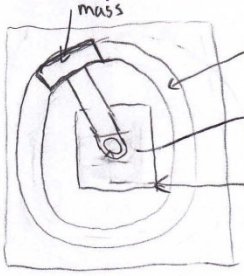
foot w/pedals (Import)



← generator (Produce)

Hand Crank

hand w/crank (Import)

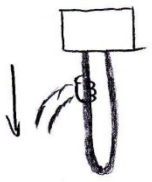


"motion absorbing device"

Scale can vary

Collect

D12·PH3·A



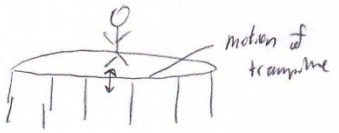
Human, hand Belt Drive Generator:

(Aid of gravity on downward stroke)

(foreground an analogy from cuckoo clocks)

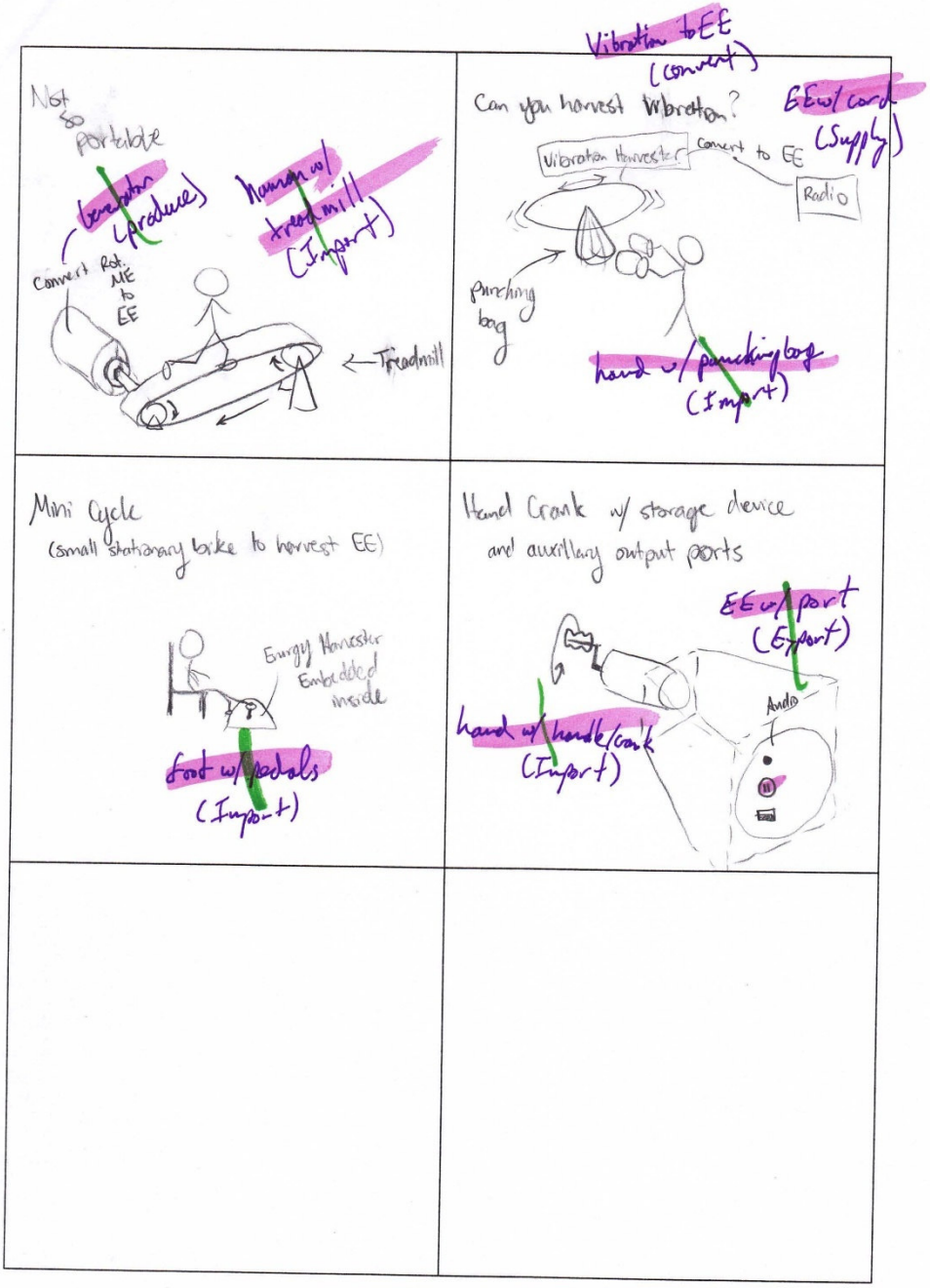
can be applied on larger scale: descending from a cliff, festival of transportation

DB-PH1-A

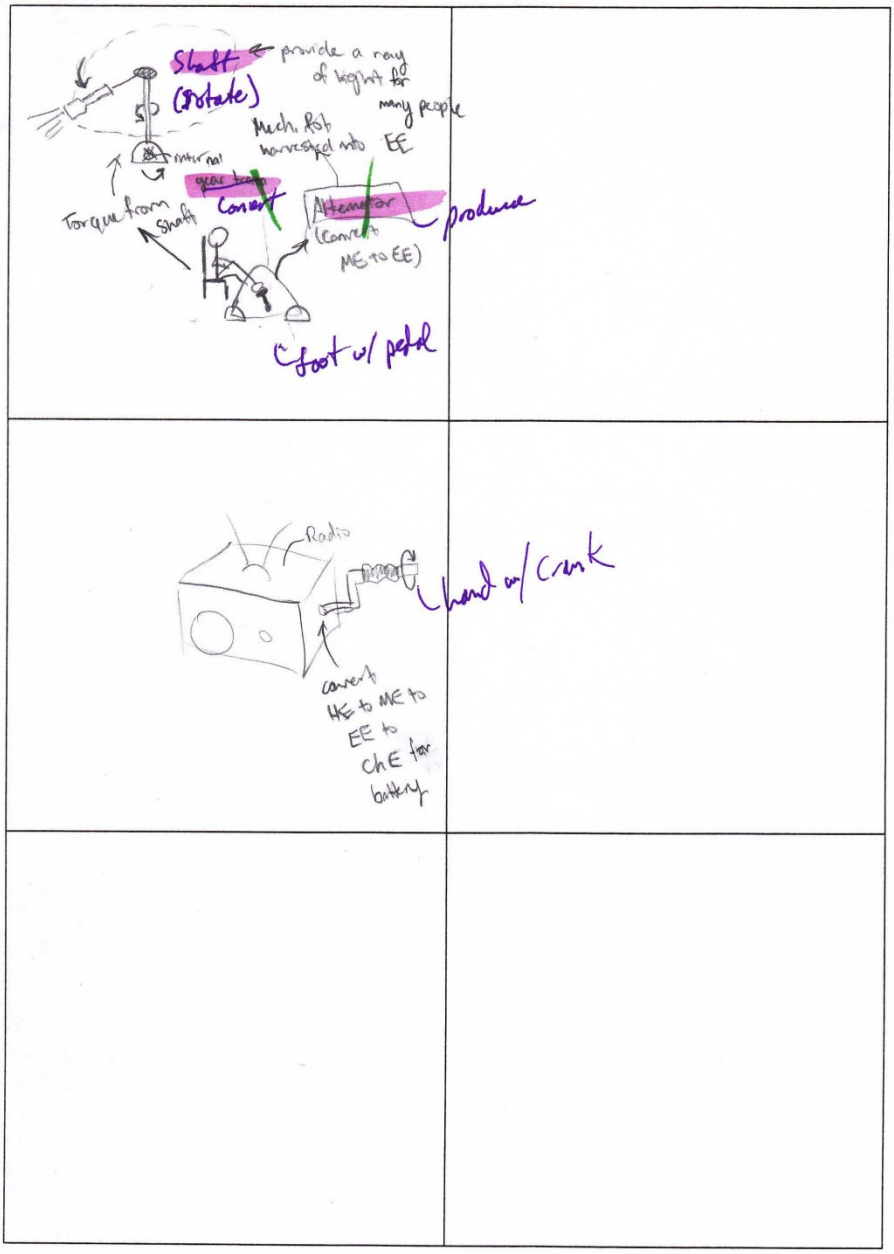
Stationary Bicycle	human w/ trampoline (Import) Trampoline: convert vertical oscillations of trampoline to electrical energy 
Pulley: Person pulls pulley system, convert rotation of pulley to electrical energy Pulley System (Convert)  hand w/ rope (Import)	Pull rope similar to how you start a lawnmower

DIS-PHS-A

Throwing a projectile like a ball, and capturing the kinetic energy ME w/ ball	Vigorously shaking a motion sensing object
compressible pedal that you push with your legs. Pedal would have some type of spring oscillate  pedal w/ foot	Capture heat, sound energy from a human body working out ME w/ walking

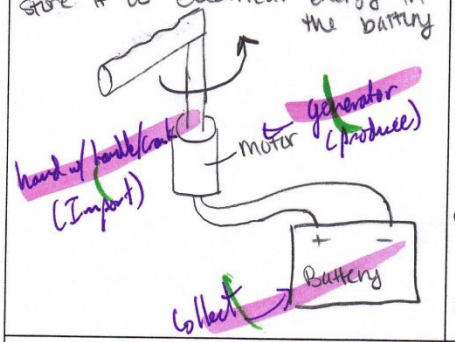

Not so portable
vibration (produce)
human w/ treadmill (Import)
Convert Rot. ME to EE
Treadmill
Vibration to EE (convert)
Can you harvest vibration?
EE w/ cord (Supply)
Vibration Harvester
Convert to EE
Radio
punching bag
hand w/ punching bag (Import)
Mini Cycle
(small stationary bike to harvest EE)
Energy Harvester Embedded inside
foot w/ pedals (Import)
Hand Crank w/ storage device and auxiliary output ports
EE w/ port (Export)
hand w/ handle/crank (Import)
Audio

DI4-PH3-A


Shaft
(rotate)
provide a way of input for many people
Mech. Rot harvested into EE
gear train
Torque from shaft
Convert
Alternator
(converts ME to EE)
produce
foot w/ pedal
Radio
hand w/ crank
convert ME to ME to EE to ChE for battery

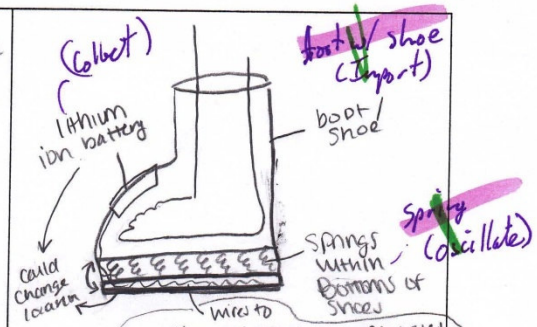
D15·PH1·A

Hand crank to change the human energy to mechanical & store it as electrical energy in the battery



(collect)

boot w/ shoe
(Import)



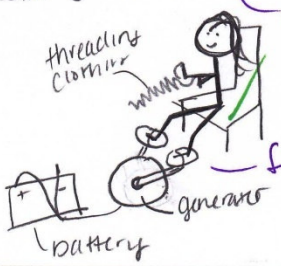
Use walking to generate energy through springs which can be stored in a detachable lithium ion battery, which can then be used @ home after work.

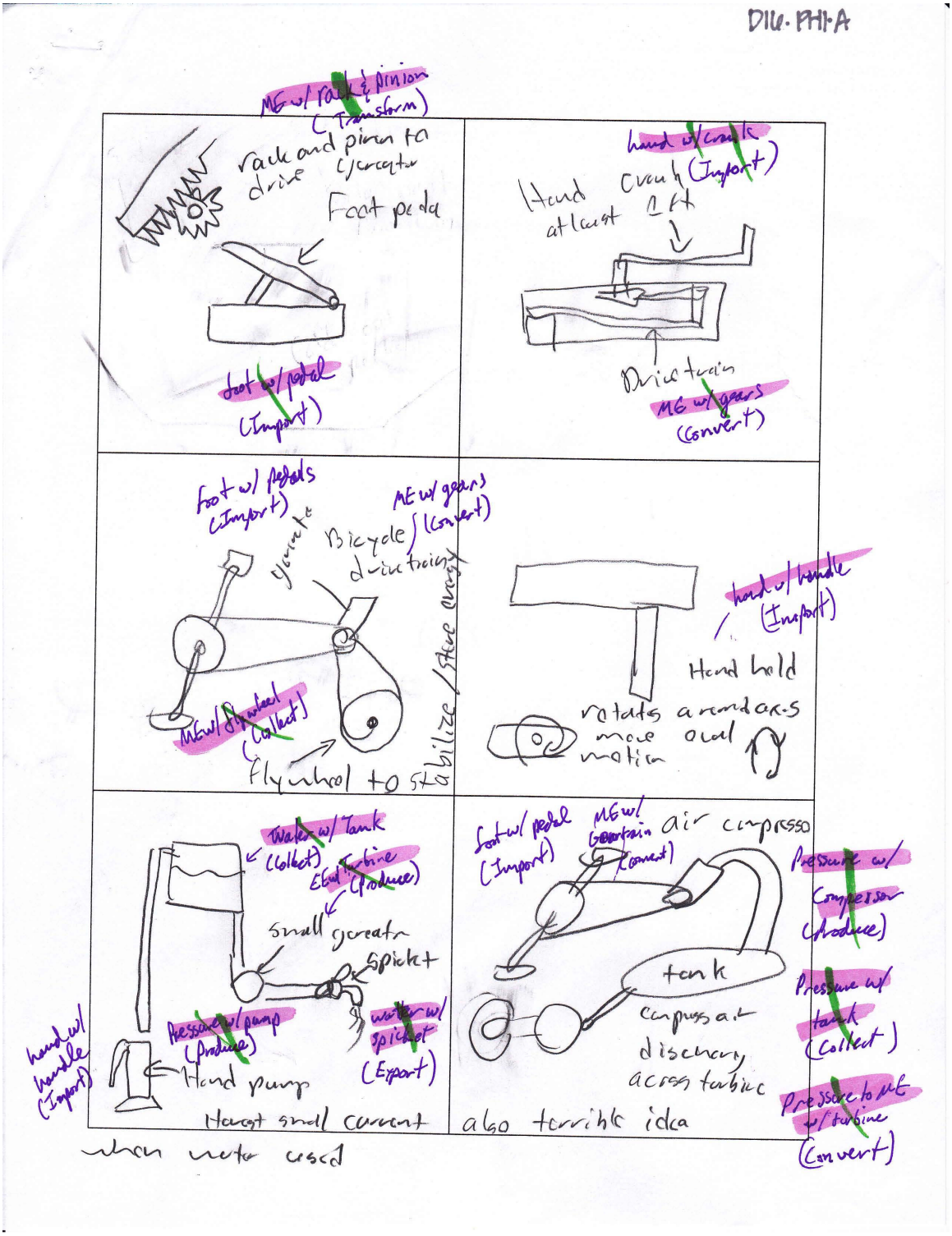
D15·PH3·A

(Bicycles could be distributed to these Countries if not already common)

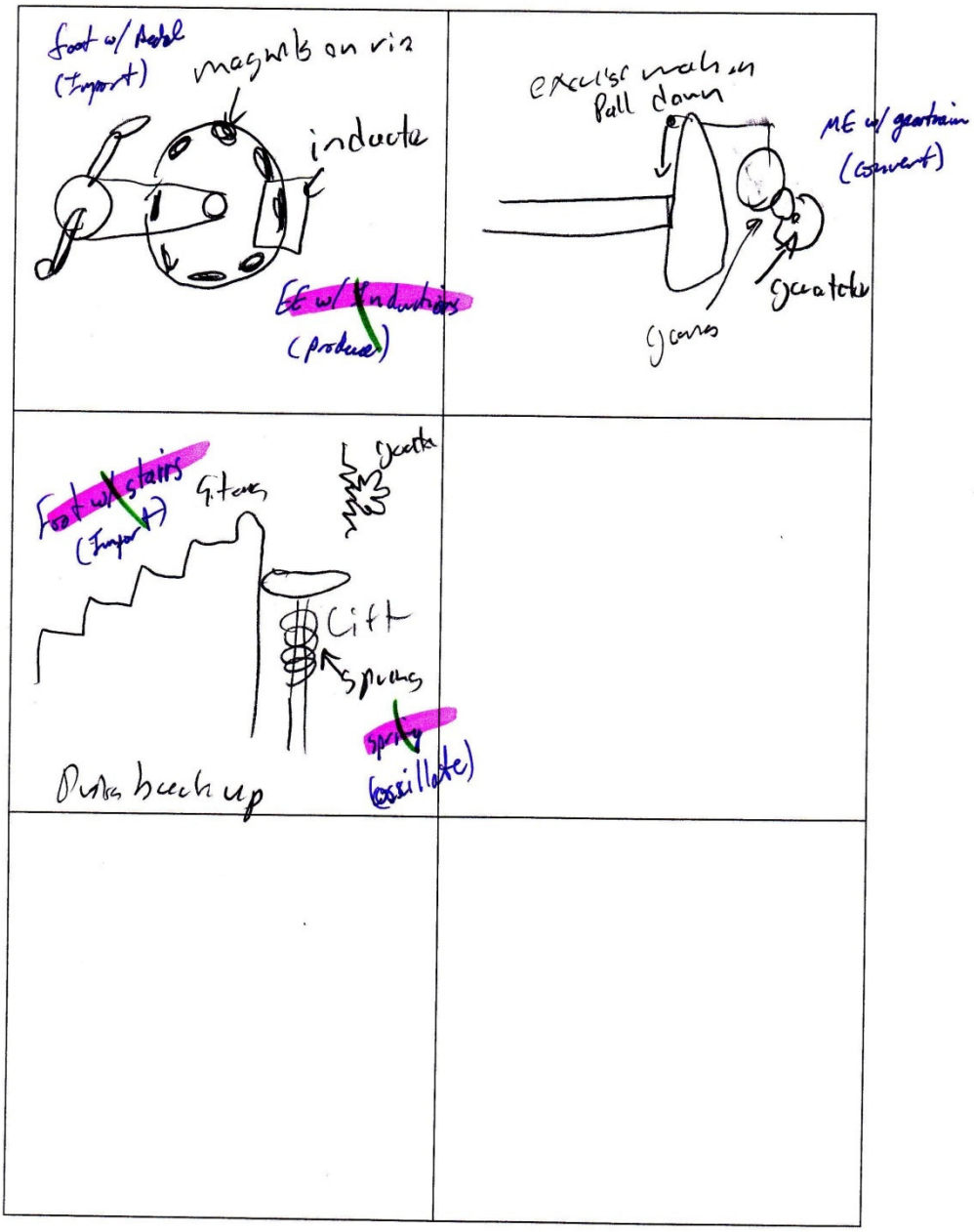
As you pedal, the generators convert the mechanical energies provided by the wheels into ~~electrical energy~~ to store into the battery

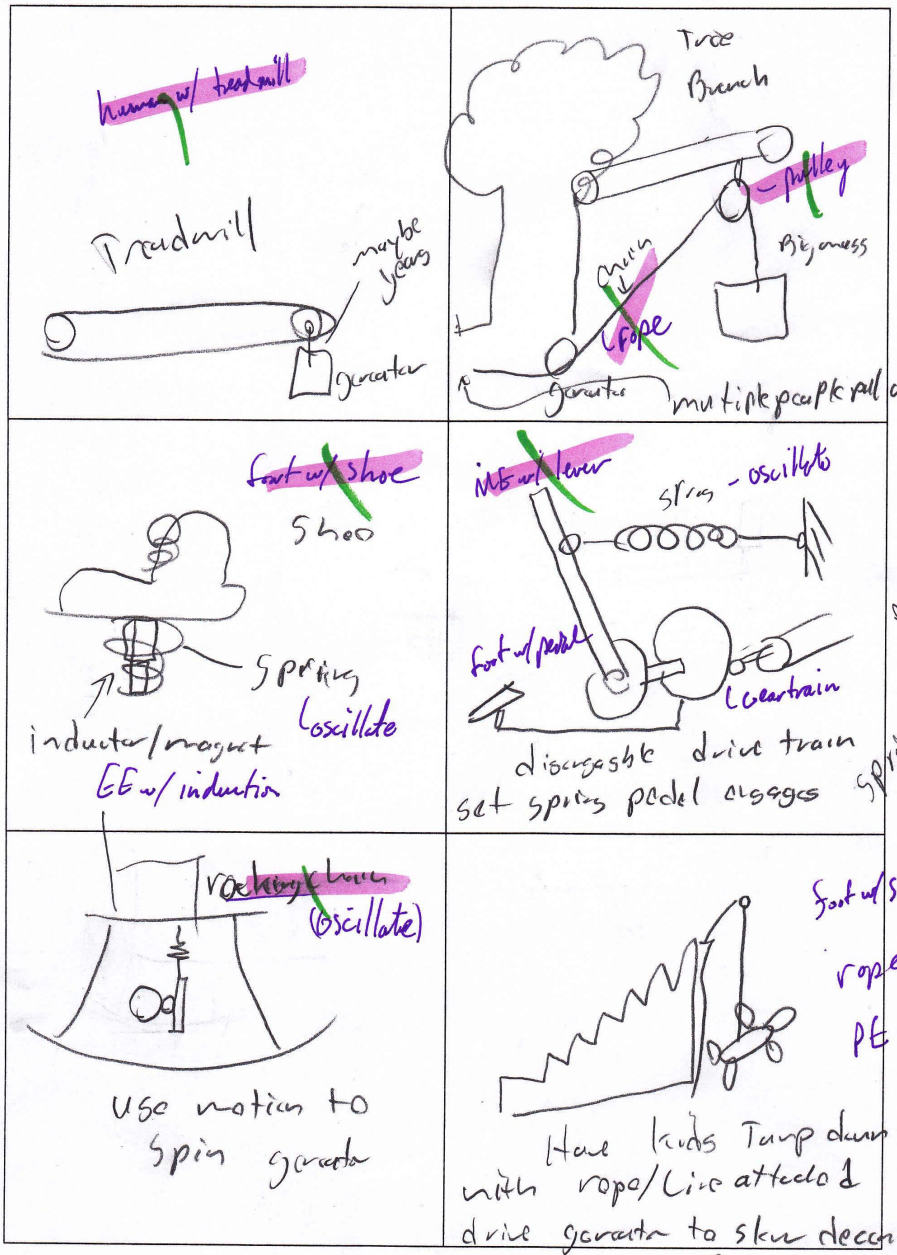
While at women (or men) cook dinner or thread clothing could sit and pedal to generate electricity to store in the battery (transport) (collect)



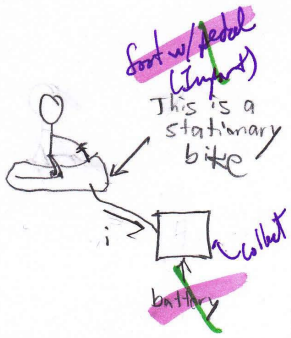

DIG. PHI-A
ME w/ rack & pinion (Transform)
rack and pinion to drive generator
Foot pedal
foot w/ pedal (Import)
hand w/crank (Import)
Hand crank at least 1 ft
Drivetrain
ME w/ gears (Convert)
foot w/ pedals (Import)
generate
Bicycle drivetrain
ME w/ gears (Convert)
ME w/ flywheel (Collect)
Flywheel to stabilize / store energy
hand w/ handle (Import)
Hand hold
rotates axes more oval motion
Water w/ Tank (Collect)
Water Turbine (Produce)
small generator
Spicket
hand w/ handle (Import)
Pressure w/ pump (Produce)
Hand pump
water w/ spicket (Export)
Hardest small current when water used
foot w/ pedal (Import)
ME w/ gearchain (Convert)
air compressor
tank
compressed air discharge across turbine
also terrible idea
Pressure w/ Compressor (Produce)
Pressure w/ tank (Collect)
Pressure to ME w/ turbine (Convert)

DIV·PHI·B


foot w/ Pedal
(Input)
magnets on rim
inductor
EE w/ Inductors
(produce)
exercise machine
Pull down
ME w/ gearbrain
(Convert)
gears
gearbox
Foot w/ stairs
(Input)
Gears
Gears
Lift
Springs
spring
(oscillate)
Pushes back up


human w/ treadmill
Treadmill
maybe gears
generator
Tree
Branch
pulley
Big mass
rope
generator
multiple people pull up
lat generator
Stair Descent
Body twist aids pulley
Back
spring returns
spins generator
foot w/ shoe
Shoe
Spring
oscillate
inductor/magnet
EE w/ induction
hub w/ lever
spring - oscillate
foot w/ pedal
Gear train
disengagable drive train
set spring pedal engages
rocking chair
(oscillate)
use motion to
spin generator
foot w/ stairs
rope
PE w/ height
Have kids Jump down
with rope/Line attached
drive generator to slow descent
charge kids for ride...

DI7·PHI·A



thermoelectric

harvest body heat to create a ΔT to make power

-affix device to skin?

heat engine (produce) convert!

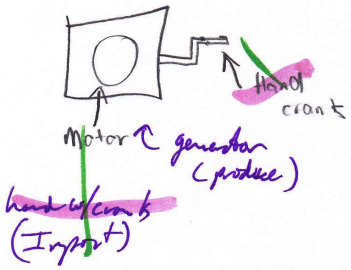
Harvest KE from oscillations made by a person moving/working + store as EE/ChE

-attached to clothing

-or shoes?

human w/ clothes (Import)

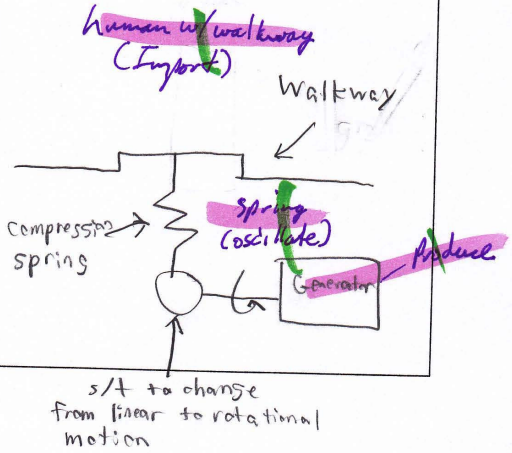
foot w/ shoes (Import)



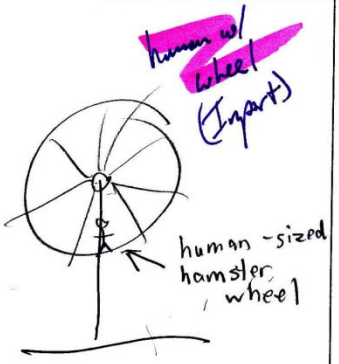
EE w/ solar panel (convert)

solar panel or equivalent attached to hat

human w/ hat (Import)



D17·PHI·B

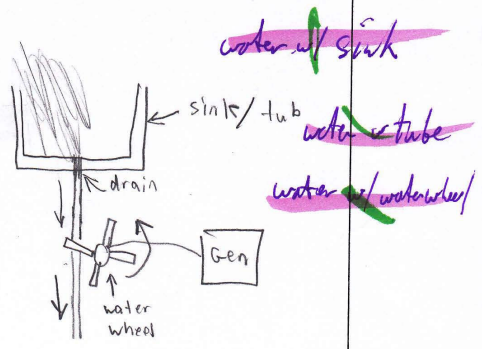


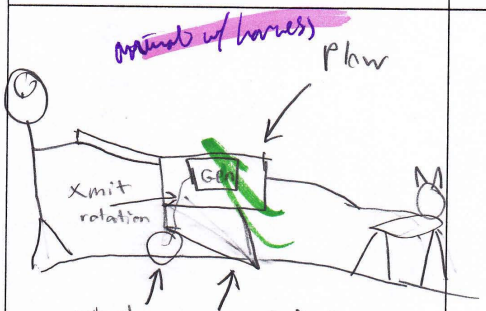
D17 · PH3 · A

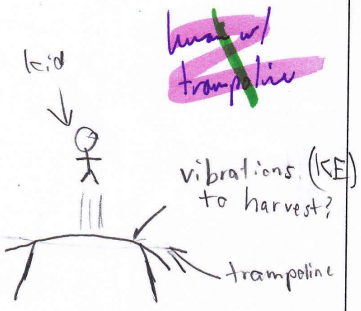
I don't feel that reading the article had any effect on my ability to come up with ideas, relative to before I had read it. (I did find the article interesting, though)

- Have a person perform a non intrusive repetitive action while walking / working where the motion is harvested & converted to electricity

ie, pulling and retracting a cord attached to a generator while swinging their arms while walking







References

Aronson, A.R, and Rindflesch, T.C., and Browne, A.C., 1994, "Exploiting a large thesaurus for information retrieval", *Proceedings of the Conference for Intelligent Multimedia Information Retrieval Systems and Management*, New York, NY.

Ball, L.J., Ormerod, T.C., and Morley, N.J., 2004, "Spontaneous analogizing in engineering design: a comparative analysis of experts and novices." *Design Studies,* 25(5), pp. 495-508.

Barker, K., Porter, B., and Clarck, P., 2001, "A Library of Generic Concepts for Composing Knowledge Bases," *International Conference on Knowledge Capture,* Victoria, Canada.

Blanchette, I., and Dunbar, K., 2000, "How Analogies are Generated: The roles of Structural and Superficial similarity," *Memory and Cognition, 28*(1), pp. 108-124.

Brin, S. and Page, L., 1998, "The Anatomy of a Large-scale Hypertextual Web Search Engine", *Proceedings of the 7th World Wide Web Conference,* Brisbane, Australia.

Bohm, M., Vucovich, J.P. and Stone, R., 2005, "Capturing Creativity: Using a Design Repository to Drive Concept Innovation", *Proceedings of DETC'05 ASME Design Engineering Technical Conferences*, Long Beach, CA.

Bohm, M. and Stone, R., 2004a, "Product Design Support: Exploring a Design Repository System", *Proceedings of IMECE'04*, Anaheim, CA.

Bohm, M., and Stone, R., 2004b, "Representing Functionality to Support Reuse: Conceptual and Supporting Functions", *Proceedings of DETC'04 ASME Design Engineering Technical Conferences*, Salt Lake City, UT

Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., and Campbell, M., 2005a, "Concept Generation from the Functional Basis of Design" *Proceedings of the International Conference on Engineering Design*. Melbourne, Australia.

Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., and Campbell, M., 2005b, "A Computational technique for Concept Generation" *Proceedings of DETC'05 ASME Design Engineering Technical Conference,* Long Beach, CA .

Cagan, J., Chan, J., Fu, K., Schunn, C., Wood, K.L., and Kotovsky, K., 2011, "On the Effective Use of Design-by-Analogy: The Influences of Analogical Distance and Commonness of Analogous Designs on Ideation Performance", *Proceedings of the International Conference on Engineering Design*. Copenhagen, Denmark.

Chakrabarti, A. and Bligh, T.P., 2001, "A Scheme for Functional Reasoning in Conceptual Design", *Design Studies,* v 22, pp.493-517.

Chan, J., Fu, K., Schunn, C., Cagan, J., Wood, K., and Kotovsky, K., 2011, "On the Benefits and Pitfalls of Analogies for Innovative Design: Ideation Performance

Based on Analogical Distance, Commonness, and Modality of Examples," ASME *Journal of Mechanical Design (JMD)*, v. 133(8), DOI: 10.1115/1.4004396.

Chiu, M., 2003, "Design Moves in Situated Design with Case-based Reasoning", *Design Studies*, v 24, pp. 1 – 25.

Chiu, I., and Shu, L. H., 2007, "Using Language as Related Stimuli for Concept Generation," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM,* v. 21(2), pp. 103-121

Chung, Y.M., and Lee, J.Y., 2001, "A Corpus-Basedproach to Comparative Evaluation of Statistical Term Association Measures," *Journal of the American Society for Information Science and Technology,* v52(4), pp. 283-296.

Cleverdon, C.W., 1967, "The Cranfield tests on index language devices," *Aslib Proceedings*, v. 19, pp. 173–192.

Copeck, T. and Szpakowicz, 2004, "Vocabulary Agreement among Model Summaries and Source Documents", *ACL Text Summarization Workshop,* Barcelona, Spain.

Dumais, S.T., 1995, "Latent Semantic Indexing (LSI): TREC-3 report," *Proceedings of TREC*, pp. 219–230.

Eastman, C., and Jansen, B., 2003, "Coverage, Relevance, and Ranking: The Impact of Query Operators on Web Search Engine Results," *ACM Transactions on Information Systems,* v. 21(4), pp. 383-411.

Falkenhainer, B.F., Forbus, K.D., and Gentner, D., 1989, "The Structure Mapping engine: Algorithm and Examples," *Artificial Intelligence*, 41(1), pp. 1-63.

Forbus, K.D., Gentner, D., and Law, K., 1994, "MAC/FAC: A model of similarity based retrieval," *Cognitive Science*, v.19, pp. 141-205.

USPTO, 2010, *General Information Concerning Patents,* http://www.uspto.gov/patents/resources/general_info_concerning_patents.pdf

Gentner, D., 1981, "Some Interesting Differences between Verbs and Nouns", *Cognition and Brain Theory,* v. 4(2), pp. 161-178.

Gentner, D., 1983, "Structure Mapping – A Theoretical Framework", *Cognitive Science,* v 7, pp. 155 – 177.

Gentner, D., and Kurtz, K.J., 2006, "Relations, Objects, and the Composition of Analogies," *Cognitive Science,* v. 30(4), pp. 609-642.

Gentner, D. and Markman, A.B., 1997, "Structure Mapping in Analogy and Similarity," *American Psychologist, 52,* pp. 45-56

Goldschmidt, G. and Weil, M., 1998, "Contents and Structure in Design Reasoning," *Design Issues*, 14(3), pp. 85-100

Hacco, E., and Shu, L. H., 2002, "Biomimetic concept generation applied to design for remanufacture," *Proceedings of the DETC 2002 ASME Design Engineering Technical Conference*, Montreal, Quebec, Canada.

Hauser, J.R., and Clausing, D., 1988, "The House of Quality," *Harvard Business Review*, v. 3, pp. 63-73.

Himmeroder, R., Lausen, G., Ludascher, B. and Schlepphorst, C., 1997, "On Declarative Semantics for Web Queries", *5th Intl. Conf. on Deductive and Object-Oriented Databases,* Montreux, Switzerland

Hirtz, J., Stone, R., McAdams, D., Szykman, S. and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts", *Research in Engineering Design,* 13(2), pp. 65-82.

Hull, D., Ait-Mokhtar, S., Chuat, M., Eisele, A., Gaussier, E., Grefenstette, G., Isabelle, P., Samuelsson, C. and Segond, F., 2001, "Language Technologies and Patent Search and Classification", *World Patent Information,* v 23, pp. 265 – 268.

Kang, I., Na, S., Kim, J., and Lee, J., 2007, "Cluster-based Patent Retrieval," *Information Processing and Management,* v. 43, pp 1173-1182.

Kitamura, Y. and Mizoguchi, R., 1998, "Functional Ontology for Functional Understanding," *Twelfth International Workshop on Qualitative Reasoning,* CapeCod, Massachusetts.

Kolodner, J.L., 1997, "Educational Implications of Analogy: A View from Case-Based Reasoning", *American Psychologist,* v 52(1), pp. 57 – 66.

Kottlowski, M., Halverson, J., and Smith, C., 2007, "A Design Review for the Research and Design of Two Robotic End-Effectors for the Los Alamos National Laboratory PDCF Disassembly Lathe", *Senior Design Project Report*, University of Texas at Austin.

Kryssanov, V.V., Tamaki, H. and Kitamura, S., 2001, "Understanding Design Fundamentals: How Synthesis and Analysis Drive Creativity, Resulting in Emergence", *Artificial Intelligence in Engineering,* v 15, pp. 329 – 342.

Kumar, R., Raghavan, P., Rajagopalan, S. and Tomkins, A., 1999, "Extracting Large-scale Knowledge Bases from the Web", *Proceedings of the 25th VLDB Conference,* Edinburgh, Scotland.

Kurfman, M., Rajan, J., Stone, R. and Wood, K., 2001, "Functional Modeling Experimental Studies *Proceedings of DETC'01 ASME Design Engineering Technical Conferences*, Pittsburgh, PA..

Kurfman, M., Rajan, J., Stone, R. and Wood, K., 2003, "Experimental Studies Assessing the Repeatability of a Functional Modeling Derivation Method," *Journal of Mechanical Design*, v 125(4), pp. 682-693.

Lochbaum, K.E., and Streeter, L.A., 1989, "Comparing and Combining the Effectiveness of Latent Semantic Indexing and the Ordinary Vector Space Model for Information Retrieval", *Information Processing & Management,* v. 25(6), pp. 665-676.

Larkey, L.S., 1999, "A Patent Search and Classification System", *Proceedings of the 4th ACM International Conference on Digital Libraries,* Berkeley, CA.

Linsey, J.S., 2007, *Design-by-Analogy and Representation in Innovative Engineering Concept Generation*, The University of Texas at Austin.

Linsey, J.S., Clauss, E.F., Kurtoglu, T., Murphy, J.T., Wood, K.L., and Markman, A.B., 2011, "An Experimental Study of Group Idea Generation Techniques: Understanding the Roles of Idea Representation and Viewing Methods", *Journal of Mechanical Design,* v 133(3).

Linsey, J.S., Green, M.G., Van Wie, M., Wood, K.L., and Stone, R., 2005, "Functional Representations in Conceptual Design: A First Study in Experimental Design and Evaluation," *Proceedings of 2005 American Society for Engineering Education Annual Conference*, Portland, Oregon.

Linsey, J.S., Murphy, J.T., Markman, A., Wood, K.L. and Kortoglu, T., 2006, "Representing Analogies: Increasing the Probability of Innovation," *ASME International Design Theory and Methods Conference*, Philadelphia, PA.

Linsey, J.S., Tseng, I., Fu, K., Cagan, J., Wood, K.L, and Schunn, C., 2010, "A Study of Design Fixation, Its Mitigation and Perception in Engineering Design Faculty," *Journal of Mechanical Design,* v. 132, 041003-1.

Liu, H. and Singh, P., 2004, "Commonsense Reasoning in and over Natural Language", *8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems,* Wellington, New Zealand.

Liu, H. and Singh, P., 2004, "ConceptNet – A Practical Commonsense Reasoning Tool-kit", *BT Technology Journal,* v. 22(4), pp. 211 – 226.

Manning, C.D., Raghavan, P., and Schutz, H., 2009, *An Introduction to Information Retrieval,* Cambridge University Press, Cambridge, England.

Markman, A. and Wood, K. L., editors, 2009, *Tools of Innovation: The Science Behind the Practical Methods that Drive New Ideas*, Oxford University Press, New York, NY.

Matlab User Guide, Matlab R2009b, Mathworks, Inc.

McAdams, D. A., Stone, R. B., and Wood, K. L., 1998, "Understanding Product Similarity Using Customer Needs," *Proceeding of the DETC'98 ASEM Design Engineering Technical Conferences,* Atlanta, Georgia.

McAdams, D. A., Stone, R. B., and Wood, K. L., 1999, "Functional Interdependence and Product Similarity Based on Customer Needs," *Research in Engineering Design*, 11, pp. 1-19.

McAdams, D., and Wood, K., 2002, "A Quantitative Similarity Metric for Design by Analogy," *ASME Journal of Mechanical Design*, v. 124(2), pp.173-182.

van der Meulen, W.A., and Janssen, P.J.F.C., 1977, "Automatic versus Manual Indexing," *Information Processing & Management,* v. 13, pp. 13-21.

Moldovan, A., Bot., R.I., and Wanka, G., 2005, "Latent Semantic Indexing for Patent Documents," *International Journal of Applied Mathematics and Computer Science,* v. 15(4), pp. 551-560.

Murdock, J., Szykman, S. and Sriram, R., 1997, "An Information Modeling Framework to Support Design Databases and Repositories *Proceedings of DETC'97 ASME Design Engineering Technical Conferences*, Sacramento, CA

Osborn, A., 1957, *Applied Imagination*, Scribner, New York, New York.

Otto, K. and Wood, K., 2001, *Product Design Techniques in Reverse Engineering and New Product Development,* Prentice Hall, Upper Saddle River, New Jersey.

"Personal Hovercraft," 2010, http://charleston-southcarolina.olx.com/personal-hovercraft-iid-736130#pics

"Polaris Sportsman 6x6 ATV", 2006, http://www.travelizmo.com/archives/000724.html

"Polaris Sportsman 500 ATV", 2003, http://www.tradebit.com/usr/manualman/pub/9002/2003-sportsman-500-polaris.jpg

Porter, M.F., 1980, "An Algorithm for Suffix Stripping," *Program*, v. 14(3), pp. 130−137.

Potter, S., Culley, S.J., Darlington, M.J., and Chawdhry, P.K., 2003, "Automatic Conceptual Design Using Experience-derived Heuristcs", *Research in Engineering Design,* v. 14, pp. 131 – 144.

Pahl, G. and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, 2nd Edition, Springer-Verlag, London, UK.

Rindflesch, T.C. and Aronson, A.R., 2002, "Semantic processing for enhanced access to biomedical knowledge", *Real World Semantic Web Applications*, IOS Press, pp. 157-172.

Rindflesch, T.C., 1996, "Natural Language Processing", *Annual Review of Applied Linguistics,* v.16, pp. 71-85.

Römer, A., Weißhahn, G. and Hacker, W., 2001, "Effort-saving product representations in design – results of a questionnaire survey," *Design Studies*, 22(6), pp. 473-490.

Rowe, B., Wood, D., Link, A., and Simoni, D., 2010, *Economic Impact Assessment of*

NIST's Text REtrieval Conference (TREC) Program, http://trec.nist.gov/pubs/2010.economic.impact.pdf

Salton, G., 1971, *The SMART Retrieval System—Experiments in Automatic Document Retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ.

Salton, G. and McGill, M.J., 1986, *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Salton, G. and Waldstein, R.K., 1978, "Term Relevance Weights in On-line Information Retrieval," *Information Processing & Management,* v. 14, pp. 29-35.

Salton, G., Wong, A., and Yang, C.S., 1975, "A Vector Space Model for Information Retrieval," *Communications of the ACM*, v. 18(11), pp. 613–620.

Schmid, H., 1994, "Probabilistic Part-of-Speech Tagging Using Decision Tree," *International Conference on New Methods in Language Processing,* Manchester, UK.

Schunn, C., Wood, K.L., Cagan, J., Chan, J., Fu, K., and Kotovsky, K., 2011, "The effects of example distance and familiarity on conceptual ideation in engineering design: Initial results," *Proceedings of 2011 NSF Engineering Research and Innovation Conference*, Atlanta, Georgia.

Shah, J.J., Vargas-Hernandez, N. and Smith, S. M., 2003, "Metrics for Measuring Ideation Effectiveness," *Design Studies,* v 24, pp. 111-134.

Shah, J. J., Kulkarni, S. V. and Vargas-Hernández, N., 2000, "Evaluation of Idea Generation Methods for Conceptual Design: Effectiveness Metrics and Design of Experiments," *Transactions of the ASME Journal of Mechanical Design*, 122, pp. 377-384.

Shimomura, Y., Tanigawa, S., Takeda, H., Umeda, Y., and Tomiyama, T., 1996, "Functional Evaluation Based on Function Content," *Proceedings of the 1996 ASME Design Theory and Methodology Conference*, Irvine, CA.

Simpson, T.W., Peplinski, J.D., Koch, P.N., and Allen, J.K., 2001, "Metamodels for computer-based engineering design: survey and recommendations", *Engineering with Computers*, v 17, pp. 129–150.

"SRN4 Hovercraft Mountbatten Class", 2000, http://en.wikipedia.org/wiki/SRN4_Hovercraft_Mountbatten_Class.jpg

Stone, R. and Wood, K., 2000a, "Development of a Functional Basis for Design", *Journal of Mechanical Design*, v122(4), pp. 359-370

Stone, R. and Wood, K., 2000b, "A Heuristic Method for Identifying Modules for Product Architectures", *Design Studies,* v 21, pp. 5 – 31.

Stone, R. B., Wood, K. L., and Crawford, R. H., 2000c, "Using Quantitative Functional Models to Develop Product Architectures," *Journal of Design Studies*, v. 21, pp. 239-260.

Szykman, S., R.D. Sriram, C. Bochenek, and J. Senfaute, 2000, "Design Repositories: Next-Generation Engineering Design Databases", *IEEE Intelligent Systems and Their Applications,* v 15(3), pp 48-55.

Szykman, S., Sriram, R.D., Bochenek, C. and Racz, J., "The NIST Design Repository Project", *Advances in Soft Computing – Engineering Design and Manufacturing,* Springer-Verlag, London, 1999.

Terpenny, J. and Mathew, D., 2004, "Modeling Environment for Function-Based Conceptual Design", *Proceedings of DETC'04 ASME International Design Engineering Technical Conferences*, Salt Lake City, UT.

Topi, H., and Lucas, W., 2005, "Mix and Match: Combining Terms and Operators for Successful Web Searches," *Information Processing and Management,* v. 41, pp. 801-817.

Trippe, A.J., 2003, "Patinformatics: Tasks to Tools," *World Patent Information,* v. 25, pp. 211-221

Tseng, Y., Lin, C., and Lin, Y., 2007, "test Mining Techniques for Patent Analysis," *Information Processing and Management,* v. 43, pp. 1216-1247.

Ullman, D.G., 2003, *The Mechanical Design Process,* McGraw Hill, Boston, MA.

Ulrich, K. T., and Eppinger, S. D., 2004, *Product Design and Development*, McGraw Hill, Boston, MA.

Umeda, Y., Ishii, M., Yoshioka, M., Shiomura, Y. and Tomiyama, T., 1996, "Supporting Conceptual Design Based on the Function-Behavior-State Modeler," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, v 10, pp.275-288.

Van der Lugt, R., 2002, "Brainsketching and How it Differs from Brainstorming," *Creativity and Innovation Management*, v. 11(1), pp. 43-54.

Van der Pol, R., 2003, "Dipe-D: A Tool for Knoledge-Based Query Formulation in Information Retrieval," *Information Retrieval,* v. 6, pp. 21-47.

Vangundy, A. B., 1988, *Techniques of Structured Problem Solving*, 2nd edition, Van Nostrand Reinhold Company, NY.

van Rijsbergen, C. J., 1979, *Information Retrieval,* Butterworth-Heinemann, Oxford, UK.

Van Wie, M.J., Rajan, P., Campbell, M.I., Stone, R.B., and Wood, K.L., 2003, "Representing Product Architecture", *Proceedings of DETC'03 ASME International Design Engineering Technical Conferences*, Chicago, IL.

Velardi, P., Pazienza, M. T. and Fasolo, M., 1991, "How to Encode Semantic Knowledge: A Method for Meaning Representation and Computer-Aided Aquisition ", *Computational Linguistics,* v 17(2), pp. 153 – 170.

Wandel, D.M., 1981, "Connecting Rod," *U.S. Patent 4,269,083*, Washington, DC.

Wang, B, and Antonsson, E., 2005, "Hierarchical Modularity: Decomposition of Function Structures with Minimal Description Length Principle", *Proceedings of DETC'05 ASME Design Engineering Technical Conferences*, Long Beach, CA.

Wen, G. Jiang, L., Wen, J., and Shadbolt, N.R., 2006, "Generating Creative Ideas through Patents," *Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence,* Guilin, China pp. 681-690.

Wright, L.W., Nardini, H.K.G., Aronson, A.R. and Rindflesch, T.C., 1999, "Hierarchical Concept Indexing of Full-text Documents in the UMLS Information Sources Map", *Journal of the American Society for Information Science*, v. 50(6), pp. 514-523

Wyllys, R.E., 1981, "Empirical and Theoretical Bases of Zipf's Law," *Library Trends,* v. 30(1), pp. 53-64.

Zipf, G.K., 1949, *Human Behavior and the Principle of Least Effort,* Addision-Wesley.

Vita

Jeremy Thomas Murphy was born in Twin Falls, Idaho on May 8th, 1978, the son of Steven L. Murphy and Nancy N. Murphy and the brother of Joshua D. Murphy and Derek S. Murphy. He obtained his Bachelor of Science in Engineering in Mechanical Engineering from Boise State University, Boise, Idaho. He pursued his graduate work at The University of Texas at Austin where he obtained his Master of Science in Mechanical Engineering and Doctorate in Philosophy in Mechanical Engineering.

Permanent address: 2307 Planters Row

Sugar Land, TX 77478

This thesis was typed by Jeremy Thomas Murphy.